В. А. Каплун,
Ю. В. Баришев,
А. В. Остапенко

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Міністерство освіти і науки України
Вінницький національний технічний університет

В. А. Каплун, Ю. В. Баришев, А. В. Остапенко

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ

Лабораторний практикум

Навчальний посібник

Вінниця
ВНТУ
2015

УДК 004.421
ББК 32.973.26-02я73
К 20

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № від)

Рецензенти:
Л. І. Тимченко, доктор технічних наук, професор
А. М. Пєтух, доктор технічних наук, професор
Д. І. Катєльніков, кандидат технічних наук, доцент

Каплун, В. А.
К 20 Технологія програмування. Лабораторний практикум : навчальний посібник / В. А. Каплун, Ю. В. Баришев, А. В. Остапенко – Вінниця: ВНТУ, 2015. – 125 с.

У навчальному посібнику розглянуто різні аспекти програмування мовою С++. У початкових розділах ґрунтовно описано синтаксис та семантику основних стандартних конструктивних компонентів мови: лексем, виразів, операторів, функцій. Значну увагу приділено різновидам типів даних, зокрема опрацюванню масивів, символьних рядків, структур тощо. Далі у посібнику даються особливості застосування покажчиків і функцій, механізм використання засобів програмування для розширення їх можливостей, даються основні поняття про структури, об'єднання даних та бітові поля, масиви структур та динамічні структури. Наведено основні поняття і принципи об’єктно-орієнтованого програмування. Кожна тема містить багато прикладів повного процесу розробки програм – від формалізації до отримання результатів. Крім того, навчальний посібник містить велику кількість контрольних питань і індивідуальних завдань для самостійного теоретичного і практичного опрацювання матеріалу.

Навчальний посібник призначений для студентів напряму підготовки 6.170101 "Безпека інформаційних і комунікаційних систем" денної та зоачної форм навчання.

УДК 004.421
ББК 32.973-018.1я73

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1

РОЗРОБКА ПРОГРАМ З ЛІНІЙНОЮ СТРУКТУРОЮ

Мета роботи

- Ознайомитись з середовищем програмування Visual C++.
- Навчитись створювати консольні додатки, компілювати програму, виконувати, налагоджувати, користуватись системою допомогии і підказками.
- Засвоїти основні символи графічних схем, які використовуються у програмах лінійної структури.
- Ознайомитись з етапами розробки програм.
- Навчитися розробляти програми з лінійною структурою, тобто програм, у яких всі оператори виконуються послідовно, один за одним.
- Навчитися формувати звіт з лабораторної роботи.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Складові мови програмування

У тексті на будь-якій природній мові можна виділити чотири основних елементи: символи, слова, словосполучення і речення. Подібні елементи містить і алгоритмічна мова, але слова називають лексемами (елементарними конструкціями), словосполучення – виразами, а речення – операторами. Лексеми утворюються із символів, вирази з лексем і символів, а оператори – з символів, виразів і лексем.

Алфавіт мови програмування (*alphabet programming language*) – це основні неподільні знаки, за допомогою яких пишуться всі тексти мовою.

Лексема (*token****)*** – елементарна конструкція, мінімальна одиниця мови, яка має самостійний сенс.

Вираз *(expression)* задає правило обчислення деякого значення.

Оператор *(operator)* задає закінчений опис деякої дії.

Кожен елемент мови визначається *синтаксисом* і *семантикою*. Синтаксичні визначення встановлюють правила побудови елементів мови, а семантика визначає їх зміст і правила використання.

Етапи створення програми

Об'єднана єдиним алгоритмом сукупність описів і операторів утворює програму на алгоритмічній мові. Для того, щоб виконати програму, потрібно перевести її на мову, зрозумілу процесору – в машинні коди. Етапи процесу підготовки програми на C/C++ наведені на рис. 1.1 [1]: редагування; препроцесорна обробка; компіляція; компонування; завантаження; виконання.

Редагування *(editing)*. Спочатку програму необхідно підготувати за допомогою одного з текстових редакторів. Це вихідний (початковий) код програми (.c, .ср, .cpp). Файл з вихідним текстом – це ще не програма, її не можна запустити і виконати.

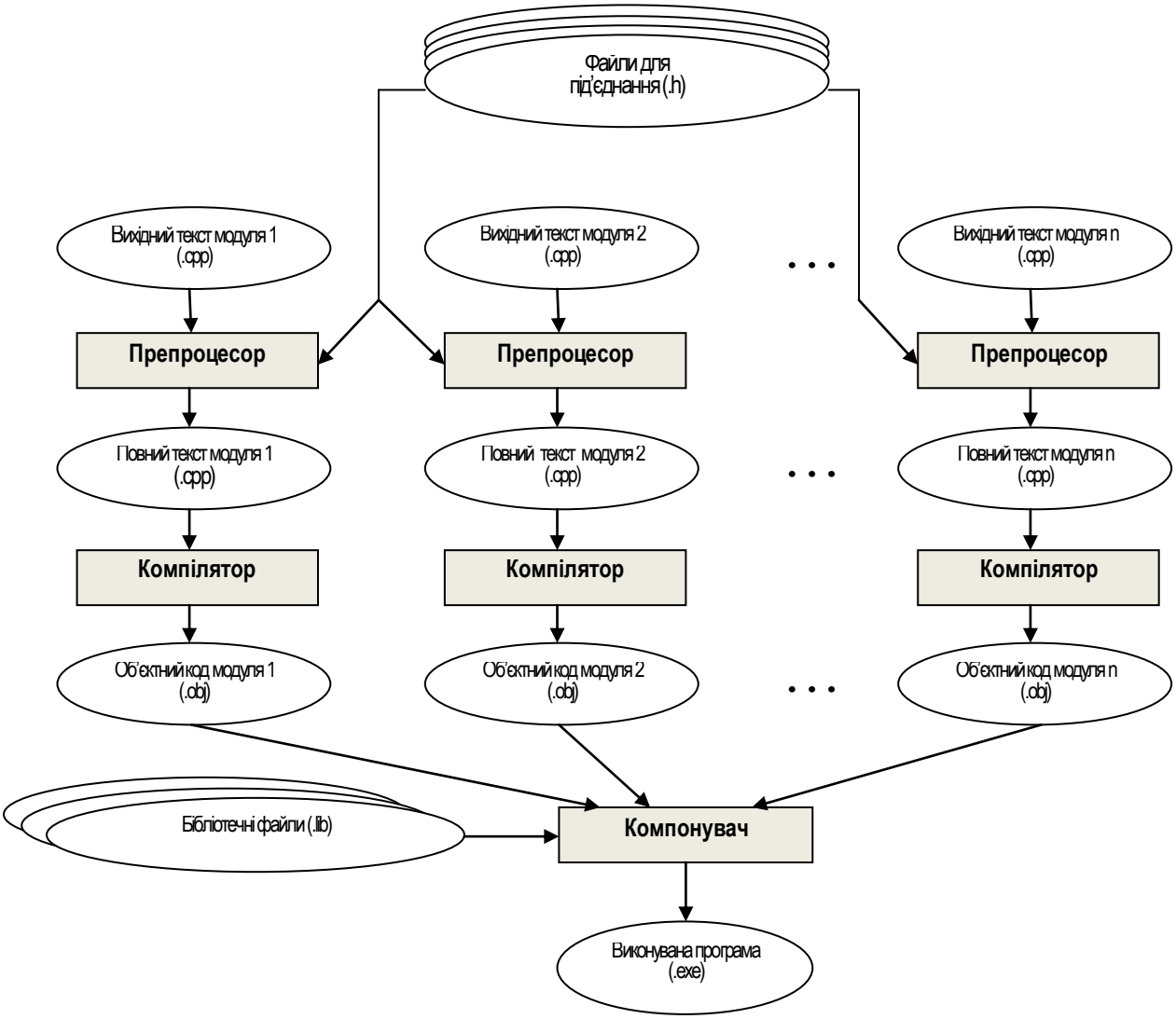


Рисунок 1.1 – Етапи процесу підготовки програми на С++

Препроцесорна обробка *(preprocessing)*. Препроцесор виконує директиви, що містяться в тексті програм (наприклад, включення в текст так званих заголовочних файлів або макровизначень).

Компіляція *(compilation)*. Одержаний повний текст програми надходить на вхід компілятора, який виділяє лексеми, а потім, на основі граматики мови програмування, розпізнає вирази і оператори, побудовані з цих лексем. При цьому компілятор виявляє синтаксичні помилки і, в разі їх відсутності, будує об'єктний модуль.

Компілятор (compiler) – програма, що перетворює текст програми на мові високого рівня на еквівалентну програму на машинній мові.

Транслятор (translator) – програма, що перетворює програму з однієї мови програмування на іншу.

Інтерпретатор (interpreter) – програма, що виконує покомандну обробку і виконання програми (без переведення її на машинну мову).

Компонування. На цьому етапі формується виконуваний модуль програми, підключаючи до об'єктного модуля інші об'єктні модулі, в тому числі ті, що містять функції бібліотек, звернення до яких є в будь-якій програмі (наприклад, для здійснення виведення на екран). Якщо програма складається з декількох вихідних файлів, вони компілюються окремо і об'єднуються на етапі компонування.

Компонувач (редактор зв'язків, лінкер – від англ. Link editor, linker) – це програма, яка виконує компонування – приймає на вхід один або декілька об'єктних модулів і збирає у виконуваний модуль.

Бібліотека в програмуванні – набір підпрограм і/або класів (об’єктів), що використовуються для розробки програмного забезпечення. У деяких мовах програмування це те саме, що модуль, в деяких – декілька модулів. Кожна бібліотека зазвичай має заголовочний файл, у якому містяться прототипи функцій, а також оголошення спеціальних типів даних і макросимволів, що використовують ці функції. З точки зору ОС та прикладного ПЗ бібліотеки поділяються на статичні та динамічні.

Статичні бібліотеки можуть бути у вигляді початкового тексту, що підключається програмістом до своєї програми на етапі написання, або у вигляді об'єктних файлів, що приєднуються (лінкуються) до виконуваної програми на етапі компіляції (у Windows такі файли мають розширення .lib, у UNIX-подібних ОС — зазвичай .a).

Динамічні бібліотеки називаються розподілюваними бібліотеками (*shared library*), або бібліотеками, що динамічно підключаються (*Dynamic Link Library*, DLL). Це окремі файли, що надають програмі набір використовуваних функцій для завантажування на етапі виконання.

Виконуваний модуль має розширення. exe і запускається на виконання звичайним чином.

Базові конструкції мови С

Алфавіт мови Сі включає :

- великі та малі літери латинської абетки;
- арабські цифри;
- пробільні символи : пробіл, символи табуляції, символ переходу на наступний рядок тощо;
- символи , . ; : ? ' ! | / \ ~ ( ) [ ] { } < > # % ^ & – + * =

Ідентифікатори використовуються для іменування різних об'єктів: змінних, констант, міток, функцій тощо. При записі ідентифікаторів можуть використовуватися великі та малі літери латинської абетки, арабські цифри та символ підкреслення. Ідентифікатор не може починатися з цифри і не може містити пробілів. Компілятор мови С розглядає літери верхнього

та нижнього регістрів як різні символи. Наприклад, кожний з наступних ідентифікаторів унікальний:

```
Sum        sum        sUm        SUM        sUM
```

Константами називають сталі величини, тобто такі, які в процесі виконання програми не змінюються. В мові Сі існує чотири типи констант: цілі, дійсні, рядкові та символьні.

1. Цілі константи бувають десятковими, вісімковими та шістнадцятковими.

Десяткова константа – послідовність десяткових цифр (від 0 до 9), яка починається не з нуля, якщо це число не нуль. Приклади: 10, 132, 1024.

Вісімкові константи починаються з символу 0, після якого розміщуються вісімкові цифри (від 0 до 7). Наприклад: 023.

Шістнадцяткові константи починаються з символів 0x або 0X, після яких розміщуються шістнадцяткові цифри (від 0 до F, можна записувати їх у верхньому чи нижньому регістрах). Наприклад : 0XF123.

2. Дійсні константи складаються з цілої частини, десяткової крапки, дробової частини, символу експоненти (e чи E) та показника степеня. Дійсні константи мають наступний формат представлення :

```
[ ціла_частина ][ . дробова_частина ][ E [-] степінь ]
```

Приклади дійсних констант: 2.2 , 220e-2, 22.E-1, .22E1.

3. Символьна константа – це один або декілька символів, які заключені в апострофи. Якщо константа складається з одного символу, вона займає в пам'яті 1 байт (тип *char*). Двосимвольні константи займають в пам'яті відповідно 2 байти (тип *int*).

Спеціальний символ	Значення
\a	звуковий сигнал
\n	перехід на наступний рядок
\r	повернення каретки
\t	горизонтальна табуляція
****	символ \
\'	символ '
\"	символ "
\?	символ ?
\0	нульовий символ
\0ddd	вісімковий код символу
\0xddd	шістнадцятковий код

Послідовності символів, які починаються з символу "\" (обернений слеш) називаються керуючими або escape-послідовностями.

4. Рядкові константи записуються як послідовності символів, заключених в подвійні лапки.

```
"Це рядковий літерал!\n"
```

Для формування рядкових констант, які займають декілька рядків тексту програми використовується символ \ (обернений слеш):

```
"Довгі рядки можна розбивати на \
частини"
```

Модифікатор const попереджає будь-які присвоювання даному об'єкту, а також інші дії, що можуть вплинути на зміну значення. Наприклад:

```
const maxint = 32767;
char *const str="Hello,P...!"; /* покажчик-константа */
char const *str2= "Hello!"; /* покажчик на константу */
```

Коментарі в С здебільшого використовуються для "документування програм" та під час їх відлагодження і виділяються за допомогою /*… */.

```
/*функція обчислює суму матриць */
```

Функція main(): з цього все починається

Усі програми, написані мовою С, повинні містити в собі хоча б одну функцію. Функція main() – вхідна точка будь-якої програмної системи, причому немає різниці, де її розміщувати. Але слід пам'ятати: якщо вона відсутня, завантажувач не зможе зібрати програму, про що буде виведене відповідне попередження. Перший оператор програми повинен розміщуватися саме в цій функції. Мінімальна програма мовою Сі має вигляд:

```
int main()
{
      return 0;
}
```

Функція починається з імені. В даному прикладі вона не має параметрів, тому за її ім'ям розташовуються порожні круглі дужки (). Далі обидві фігурні дужки {...} позначають блок або складений оператор, з яким ми працюватимемо, як з єдиним цілим.

Базові типи даних

Базові типи даних Сі можна перерахувати у такій послідовності:

char **–** *символ*

Тип може використовуватися для зберігання літери, цифри або іншого символу з множини символів ASCII. Значенням об'єкта типу *char* є код символу. Тип *char* інтерпретується як однобайтове ціле від -128 до 127.

int **–** *ціле*

Цілі числа у діапазоні від -32768 до 32767. Як різновиди цілих чисел, у деяких версіях компіляторів існують ***short*** – *коротке ціле* (2 байти) та ***long*** (4 байти) – *довге ціле*. Розмірність цих типів може коливатися. Гарантовано лише, що співвідношення розмірності є наступним: *short <= int <=long*.

float **–** *число з плаваючою комою одинарної точності*

Тип призначений для зберігання дійсних чисел. Може представляти числа як у фіксованому форматі (наприклад, число π- 3.14159), так і в експоненціальній формі – 3.4E+8.

double **–** *число з плаваючою комою подвійної точності*

Має значно більший діапазон значень, порівняно з типом *float*, а саме $\pm(1.7 \cdot 10^{-308} \ldots 1.7 \cdot 10^{308})$.

Крім того, цілі типи *char, short, int, long* можуть використовуватися з модифікаторами *signed* (із знаком) та *unsigned* (без знаку). Цілі без знаку (*unsigned*) не можуть набувати від'ємних значень, на відміну від знакових цілих (*signed*). За рахунок цього дещо розширюється діапазон можливих додатних значень типу.

Перетворення типів

Компілятор С виконує автоматичне перетворення типів даних, особливо в математичних виразах, коли найчастіше цілочисельний тип перетворюється у тип з плаваючою комою. При цьому значення типу *char* та *int* в арифметичних виразах змішуються: кожний з таких символів автоматично перетворюється в ціле. Взагалі, якщо операнди мають різні типи, перед тим, як виконати операцію, молодший тип "підтягується" до старшого. Результат – старшого типу:

– *char* та *short* перетворюються в *int;*
– *float* перетворюється в *double*;
– якщо один з операндів *long double*, то і другий перетворюється в *long double*;
– якщо один з операндів *long*, тоді другий перетворюється відповідно до того ж типу, і результат буде *long*;
– якщо один з операндів *unsigned*, тоді другий перетворюється відповідно до того ж типу, і результат буде *unsigned*.

Але, окрім цього, в С є можливість і примусового перетворення типу, щоб дозволити явно конвертувати (перетворювати) значення одного типу даних в інший. Загальний синтаксис перетворення типу має два варіанти:

```
1)(новий_тип) вираз ;
2) новий_тип (вираз) ;
```

Наприклад,

```
char letter = 'a';
int nasc = int (letter);
long iasc = (long) letter;
```

Приклад розробки програми з лінійною структурою

Задача: Розробити програму, яка переводить температуру у градусах по Фаренгейту у градуси Цельсія за формулою:

$$C = \frac{5}{9}(F - 32),$$

де C – температура за Цельсієм, F – температура за Фаренгейтом.

Формалізація задачі

1. Вхідні дані: F – температура за Фаренгейтом.
2. Вихідні дані: C – температура за Цельсієм.
3. Типи даних. Оскільки значення температур за Цельсієм і за Фаренгейтом можуть бути і цілими, і дійсними, доцільно обрати тип *float.*
4. Перевірка правильності. Для перевірки правильності обчислень, слід підготувати декілька контрольних прикладів.

Словесний алгоритм

1. Початок. { Перейти до п.2. }
2. Оголошення змінних: C, F, X – дійсні,. { Перейти до п3.. }
3. Видати запрошення для введення F. { Перейти до п.4. }

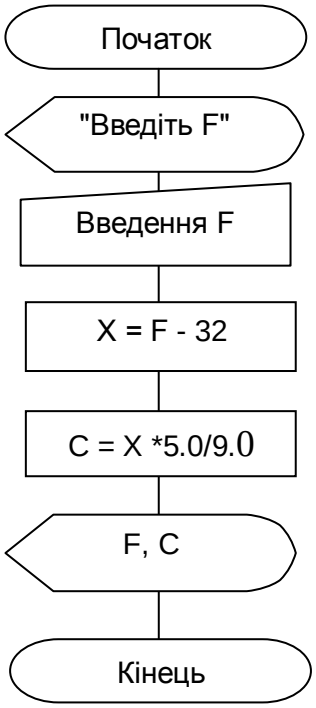
4. Ввести F. { Перейти до п.5. }
5. Обчислити X = F – 32. { Перейти до п.6. }
6. Обчислити C = X * 5.0 / 9.0. { Перейти до п.7. }
7. Вивести на екран значення F і C. { Перейти до п.8. }
8. Кінець.

Лістинг програми

```
#include <conio.h>
#include <locale.h>

int main()
{
  // оператор для встановлення російської мови
  setlocale(LC_ALL, "Russian");
  // оголошення змінних
  float F, C, X;
  printf("\n\tВведіть  F ... ");
  // запрошення на введення даних
  scanf_s("%f",&F);
  // введення значення
  X=F-32;
  C=X*5/9;
  printf("%30s%f%30s%f",
  "\n\tТемпература за Фаренгейтом: F = ",F,
  "\n\tТемпература за Цельсієм:    C = ",C);
  // очікування натискання клавіші
  // для затримки екрану
  _getch();
  return 0;
}
```

Схема роботи програми



Результати роботи програми

№	Програма	Контрольний приклад
1	Введіть F ... 225 Температура за Форенгейтом: F = 225,000000 Температура за Сельсієм: C = 107,222221	F = 225 C = 107,2
2	Введіть F ... 24,5 Температура за Форенгейтом: F = 24,500000 Температура за Сельсієм: C = -4,166667	F = 24,5 C = -4,2
3	Введіть F ... -35,2 Температура за Форенгейтом: F = -35,200001 Температура за Сельсієм: C = -37,333332	F = -35,2 C = -37,3

Висновки

1. Засвоєно основні правила роботи у середовищі Visual Studio: порядок створення проектів, використання підказок, способи налагодження.
2. Засвоєно використання таких функцій:
   - main() – головна функція програми;
   - getch() – очікування введення символу;
   - printf() – форматоване виведення на екран;
   - scanf() – форматоване введення з клавіатури ;
   - setlocale() – встановлення потрібної мови у програмах;
3. Отримано практичні навички із застосування таких базових понять: оператори присвоєння; основні типи даних; порядок математичних операцій у виразі; оператор препроцесором обробки #include.
4. Засвоєно розробку схем лінійних програм.

Порядок виконання роботи

1. Ознайомитись з середовищем програмування Visual Studio:
   - структура робочого стола середовища та головного вікна;
   - структура проектів у Visual Studio C++;
   - створення консольного додатку та його складових;
   - компілювання програми, протокол компіляції;
   - запуск програми на виконання, способи налагодження.
2. Ознайомитись з документом "Графічне представлення алгоритмів" (Додаток А), зокрема з символами для побудови схем лінійних програм:
   - символи процесу;
   - символи початку і кінця програми;
   - символи введення даних та виведення на дисплей.
3. Розробити в середовищі Visual C++ програму, наведену у прикладі далі, ознайомившись на наведеному прикладі з порядком виконання завдання і оформлення документації за результатами всіх етапів розробки.
4. Здійснити розробку програми згідно з індивідуальним завданням і підготувати звіт (для кожної задачі). Звіт повинен містити складові:
   - титульний лист (додаток Б);
   - віріант і завдання;
   - формалізація задачі та словесний алгоритм;
   - схема роботи програми;
   - лістинг програми;
   - результати виконання (не менше 3-х контрольних прикладів);
   - висновки.

Варіанти індивідуальних завдань

Варіант	Завдання А	Завдання Б
1	$z_1 = 2\sin^2(3\pi - 2\alpha) - \cos^2(5\pi - 2\alpha)$	$M = e^{-cx}\,\dfrac{x + \sqrt[3]{x+a}}{x - \sqrt{\lvert x-b\rvert}}$,
2	$z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\dfrac{\pi}{4} + 2\alpha\right)$	$F = e^{2x}\ln(a+x) - b^{3x}\ln\lvert x-b\rvert$,
3	$z_1 = \dfrac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$	$z_2 = \dfrac{1}{\sqrt{a} + \sqrt{2}}$
4	$z_1 = 1 - \dfrac{1}{4}\sin^2 2\alpha + \cos 2\alpha$	$a = y + \dfrac{x}{y^3 + \left\lvert \dfrac{x^2}{y + \sqrt[3]{x^2}} \right\rvert}$
5	$z_1 = \cos^2\left(\dfrac{3}{8}\pi - \dfrac{\alpha}{4}\right) - \cos^2\left(\dfrac{11}{8}\pi - \dfrac{\alpha}{4}\right)$	$z_2 = \dfrac{4 - a^2}{2}$
6	$z_1 = \cos^4 x + \sin^2 y + \dfrac{1}{4}\sin^2 2x - 1$	$z_2 = \sqrt{\dfrac{x+3}{x-3}}$
7	$z_1 = (\cos\alpha + \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$	$z_2 = \dfrac{\sqrt{m} - \sqrt{n}}{m}$
8	$z_2 = -4\sin^2\dfrac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta)$	$s = 1 + x + \dfrac{x^2}{2!} + \dfrac{x^3}{3!} + \dfrac{x^4}{4!}$
9	$z_1 = \dfrac{\sin^2\left(\dfrac{\pi}{2} - 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$	$z_1 = \dfrac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}$
10	$z_1 = \dfrac{1 - 2\sin^2\alpha}{1 + 2\sin\alpha}$	$S = 1 + y - \dfrac{(y-x)^2}{2} + \dfrac{\lvert y-x\rvert}{3}$,
11	$z_1 = \dfrac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \dfrac{\cos 2\alpha}{1 + \cos 2\alpha}$	$z_1 = \left(\dfrac{1 + a + a^2}{2a + a^2} + 2 - \dfrac{1 - a + a^2}{2a - a^2}\right)^{-1}(5 - 2a^2)$
12	$z_1 = \dfrac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}$	$z_1 = \left(\dfrac{a+2}{\sqrt{2a}} - \dfrac{a}{\sqrt{2a} + 2} - \dfrac{2}{a - \sqrt{2a}}\right) \cdot \dfrac{\sqrt{a} - \sqrt{2}}{a+2}$
13	$z_1 = \dfrac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$	$z_1 = \dfrac{\sqrt{(3m+2)^2 - 24m}}{3\sqrt{m} - \dfrac{2}{\sqrt{m}}}$
14	$z_2 = \dfrac{1 + \sin 2\beta}{\cos 2\beta}$	$z_1 = \dfrac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x-1)\sqrt{x^2 - 9}}$
15	$z_2 = \dfrac{1 - tg\,\alpha}{1 + tg\,\alpha}$	$z_1 = \dfrac{\sqrt{2b + \sqrt{2b^2 - 4}}}{\sqrt{b^2 - 4} + b + 2}$

Рекомендації

1. Вибирайте тип змінних з урахуванням діапазону і потрібної точності представлення даних.
2. Задавайте змінним імена, що відображають їх призначення.
3. Введення з клавіатури слід попереджати запрошенням. Для контролю відразу ж після введення слід вивести початкові дані на екран.
4. До запуску програми слід підготувати декілька контрольних прикладів для перевірки правильності обчислень. Крім того, слід підготувати декілька неправильних прикладів, щоб перевірити реакцію програми на неправильні початкові дані.
5. При записі виразів слід звертати увагу на пріоритет операцій.
6. У функціях *printf()* і *scanf()* для кожної змінної необхідно вказати специфікацію формату, що відповідає її типу. Не забувайте, що в *scanf()* передається адреса змінної, а не її значення.
7. При використанні стандартних функцій необхідно за допомогою директиви *#include ...* підключити до програми відповідні заголовочні файли. Встановити, яка бібліотека потрібна, можна за допомогою довідкової системи.

Контрольні питання

1. Які середовища програмування мовою С (С++) ви знаєте?
2. Охарактеризуйте коротко переваги і недоліка середовища програмування Visual C?
3. У чому різниця між інтерпретатором і компілятором?
4. Як відбувається компіляція вихідного коду програми?
5. Для чого існує компонувач?
6. Які види бібліотек ви знаєте? В чому різниця між ними?
7. Наведіть основні типи даних, наведіть приклади.
8. Який порядок виконання операцій у складних виразах у програмах?
9. Яким чином здійснюються перетворення типів у програмах мовою С? Наведіть основні правила перетворення типів.
10. Яка головна функція у програмах мовою С?
11. Яка послідовність дій при розробці програми?
12. Наведіть основні символи для виконання лінійних схем виконання програм і поясніть їх.
13. Які ви знаєте функції для введення інформації з клавіатури і виведення на екран?
14. Як можна задати формат виведення інформації на екран?
15. Наведіть і поясніть різні види опраторів присвоєння.

ЛАБОРАТОРНА РОБОТА №2

РОЗРОБКА СТУКТУРОВАНИХ ПРОГРАМ

Мета роботи

- Засвоїти способи застосування умовних операцій *if-else*.
- Навчитись використовувати оператор вибору *switch*.
- Оволодіти навичками щодо використання операторів різних видів операторів циклу: цикли з параметрами *for*, цикли з умовою *while*, цикли з післяумовою *do-while*.
- Дослідити роботу операторів керування: *break, continue, return, goto.*

ТЕОРЕТИЧНІ ВІДОМОСТІ

Синтаксис основних операторів

Структура вибору ***if***	**if** (<умова>){
	  <**послідовність_операторів**>;
	}

Структура вибору ***if/else***	**if** (<умова>){
	  <послідовність_операторів_1>;
	}
	else {
	  <послідовність_операторів_2>;
	}

Структура вибору ***switch***	**switch** (<вираз цілого типу>) **{**
	  case <значення_1>:
	  <послідовність_операторів_1>;
	  break;
	  ... ...
	  case <значення_n>:
	  <послідовність_операторів_n>;
	  break;
	  [**default**:
	  <послідовність_операторів_n+1>;]
	}

Оператор повторення (циклу) ***while***	**while** (<логічний вираз>) {
	  <послідовність_операторів>;
	}

Оператор повторення (циклу) ***for***	**for** ([ініціалізація];[логічний вираз];
	[нове_значення]){
	  <послідовність_операторів>;
	}

Оператор повторення (циклу) ***do while***	**do** { <послідовність_операторів>; }**while** (<логічний вираз>);
Оператор переходу ***goto***	**goto** <мітка>; /* ... */ <мітка> : <послідовність_операторів>;
Оператор розриву	**break;**
Оператор продовження	**continue;**

Розгалужена та циклічна структури алгоритмів

Розгалужена конфігурація алгоритму містить в собі як послідовності, так і розпаралелення послідовностей. Використовується, коли, залежно від умови, потрібно виконати ту чи іншу дію (рис. 2.1, а), або здійснити обхід, якщо одна гілка не містить жодних дій (рис. 2.1, б), здійснити множинний вибір, коли умова має більш, ніж три можливих варіанти (рис. 2.2)

Рисунок 2.1 – Приклади зображення на схемах операторів умови

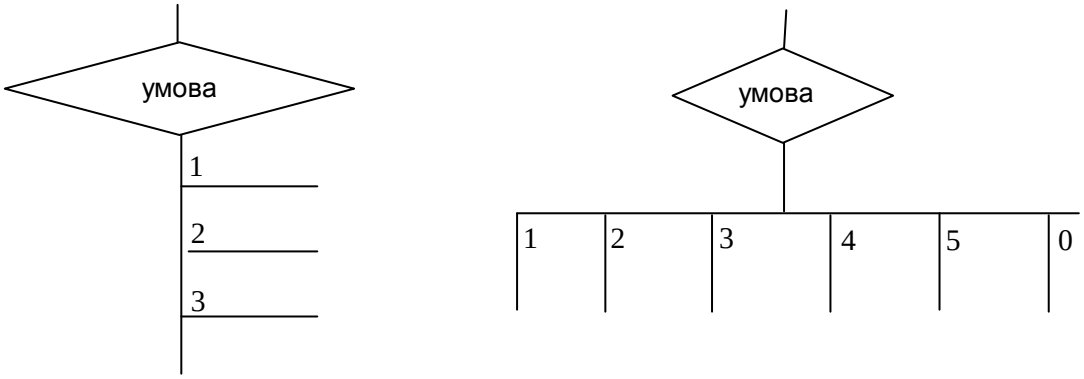


Рисунок 2.2 – Зображення на схемах операторів вибору

Циклічна структура використовується при необхідності виконувати деякі дії декілька разів. Можливе виконання циклу ***do-while,*** циклу ***while,*** циклу за параметром ***for*** (рис. 2.3).

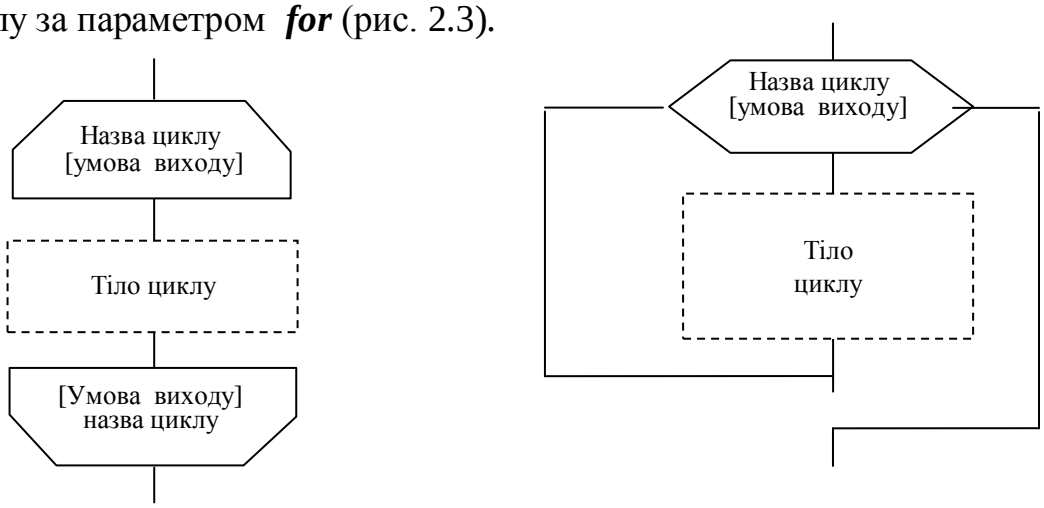


Рисунок 2.3 – Зображення на схемах операторів циклу

Приклад розробки програми з разгалуженням

Задача. Розробити програму для виведення на таблиці значень функції *y=f(x)* для аргумента *x*, що змінюється у межах [*Хпоч, Хкін*] із заданим кроком *h*.

$$y=\begin{cases} x, & якщо \quad x<0 \\ t\cdot x, & якщо \quad 0\le x\le 10 \\ x^2, & якщи \quad x>10 \end{cases}$$

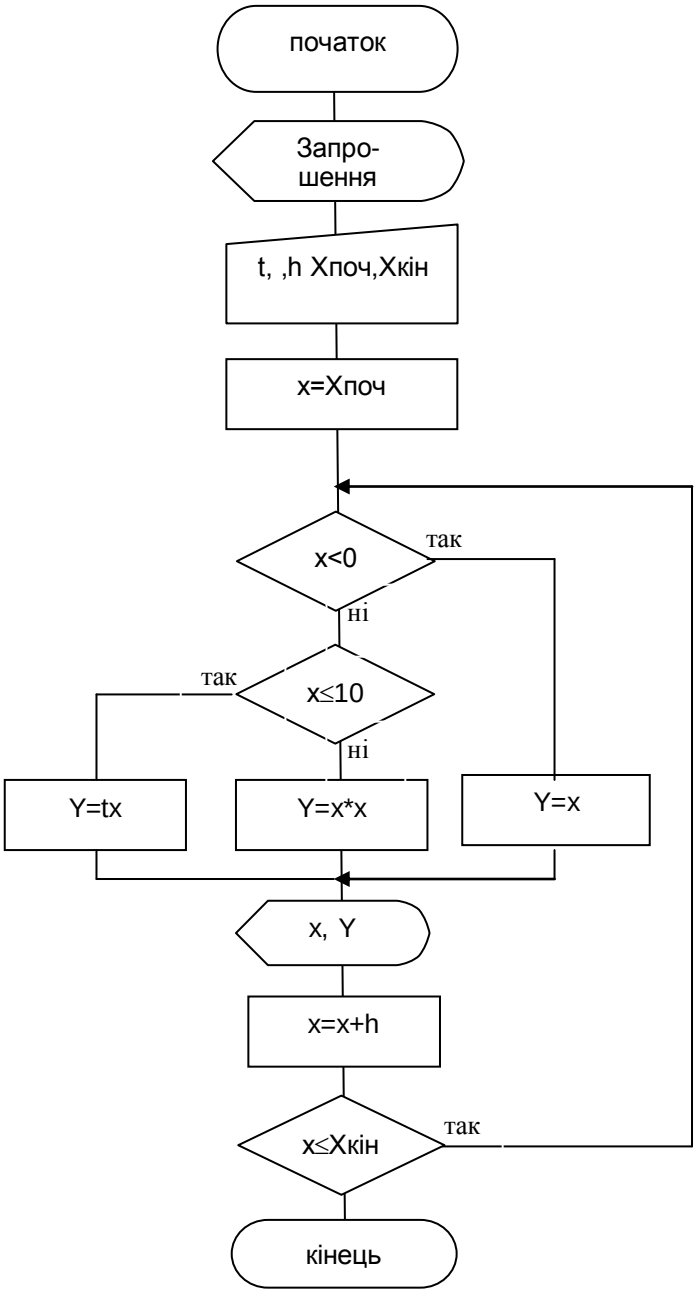
Формалізація задачі і алгоритм

Вхідні дані: *t* – ціле число, *Хпоч, Хкін, h* – дійсні числа.

1. Ввести початкові дані: *t, Хпоч, Хкін, h*. {2}
2. Надрукувати шапку таблиці. {3}
3. Взяти перше зі значень аргумента: *x=Хпоч*. {4}
4. Визначити, якому з інтервалів у формулі воно належить. {5}
5. Розрахувати значення *y*. {6}
6. Вивести рядок у таблицю. {7}
7. Збільшити *x* на *h*. {8}
8. Якщо *x>Хкін*, то {4}, інакше {9}
9. Кінець розрахунків.

Вихідні дані: значення *y* у точках.

Схема роботи програми



Лістинг програми

```
#include "stdafx.h"
#include <locale.h>
#include <conio.h>
int main()
{
  setlocale(0,"");
  int t;
  float X0,Xk,x,y,h ;
  printf("Введ1ть дан1:\n");
  printf("t = ");
  scanf_s("%d",&t);
  printf("\nВведlть Хпоч,"
         "Хк1н, h ...  ");
  scanf_s("%f %f %f", &X0,&Xk,&h);
  printf("\nІнтервал [%3.2f, %3.2f]\n Крок h = %3.2f\n", X0,Xk,h);
  printf("\n----------------------\n|     x    |     y    |");
  printf("\n|----------|---------|");
  x=X0;
  while (x<Xk)
```

```
  {   if (x<0) y=x;
      else
            if (x<=10) y=t*x;
            else y=x*x;
      printf("\n|  %7.2f |  %7.2f |",x,y);
      x+=h;
  }
  printf("\n------------------------");
  return 0;
}
```

```
Введіть початкові дані:
t = 10

Введіть Хпоч, Хкін, h ...  -10 50 5

Введено інтервал  [-10,00, 50,00]
Крок h = 5,00

------------------------
|     x     |     y     |
|-----------|-----------|
|   -10,00  |   -10,00  |
|    -5,00  |    -5,00  |
|     0,00  |     0,00  |
|     5,00  |    50,00  |
|    10,00  |   100,00  |
|    15,00  |   225,00  |
|    20,00  |   400,00  |
|    25,00  |   625,00  |
```

Приклад розробки задачі з використанням оператора вибору

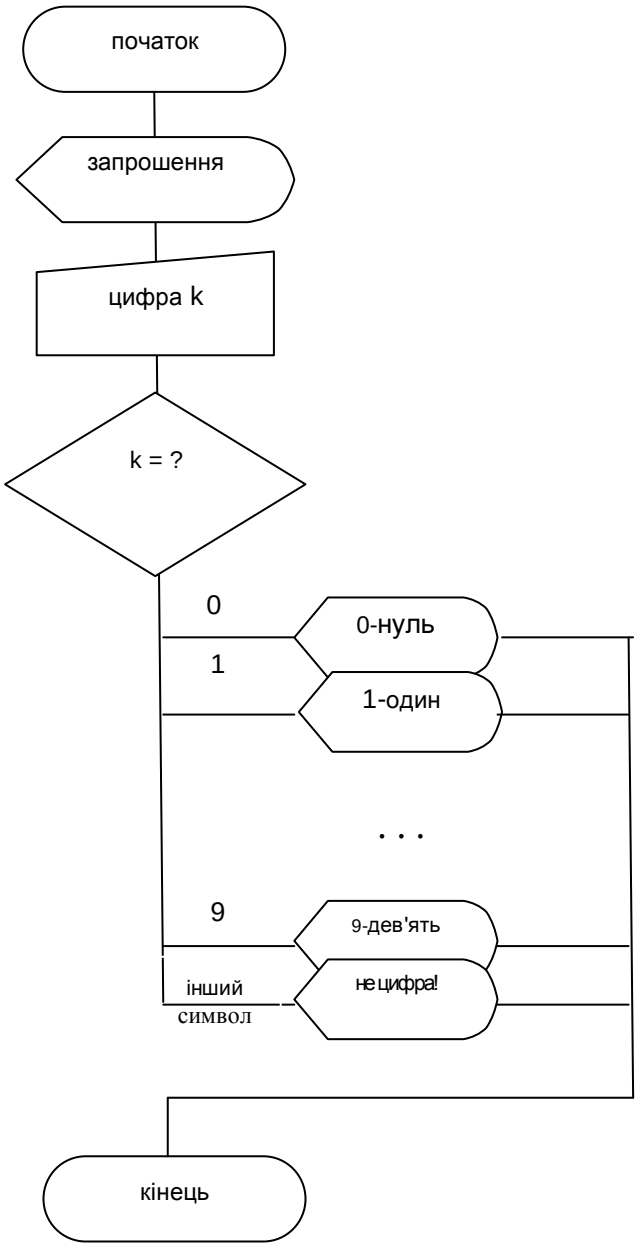
Задача. Написати програму, що визначає, яка з цифрових клавіш була натиснута.

Формалізація задачі

Вхід: цифра. Тип даних – символ.
1. Вивести запрошення на введення цифри
2. Перевірити, чи це є цифра.
3. Якщо цифра, то {4}, інакше {5}.
4. Вивести рядок з назвою відповідної цифри {6}.
5. Вивести рядок про помилку {6}.
6. Завершення програми.

Вихід: рядок з назвою цифри.

Схема роботи програми



Лістинг програми

```
#include <locale.h>
#include <conio.h>
#include "stdafx.h"
int main()
{ setlocale(0,"");
  int ch;
  printf("\nВведіть цифру ");
  scanf_s("%d",&ch);
  switch(ch)
  {    case 0: printf("\n %d%s",ch,
       " – нуль"); break;
   case 1: printf("\n %d%s",ch,
      " – один"); break;
   case 2: printf("\n %d%s",ch,
       " – два"); break;
   case 3: printf("\n %d%s",ch,
       " – три"); break;
   case 4: printf("\n %d%s",ch,
       " – чотири"); break;
   case 5: printf("\n %d%s",ch,
       " – п'ять"); break;
   case 6: printf("\n %d%s",ch,
```

```
            " – ш1сть"); break;
   case 7: printf("\n %d%s",ch,
        " – с1м"); break;
   case 8: printf("\n %d%s",ch,
        " – в1с1м"); break;
   case 9: printf("\n %d%s",ch,
        " – дев'ять"); break;
   default :
        printf("\n НЕ ЦИФРА!!!");
        ch='*'; break;
 }
_getch();
return 0;
}
```

Результати виконання програми

```
Введ1ть цифру ...  8
 8 – в1с1м
```

```
Введ1ть цифру ...  A
 НЕ ЦИФРА!!!
```

Приклад розробки програми з використанням циклів

Задача. Обчислити значення виразу $\prod_{i=1}^{25} \frac{i!}{i^2+2i+3}$.

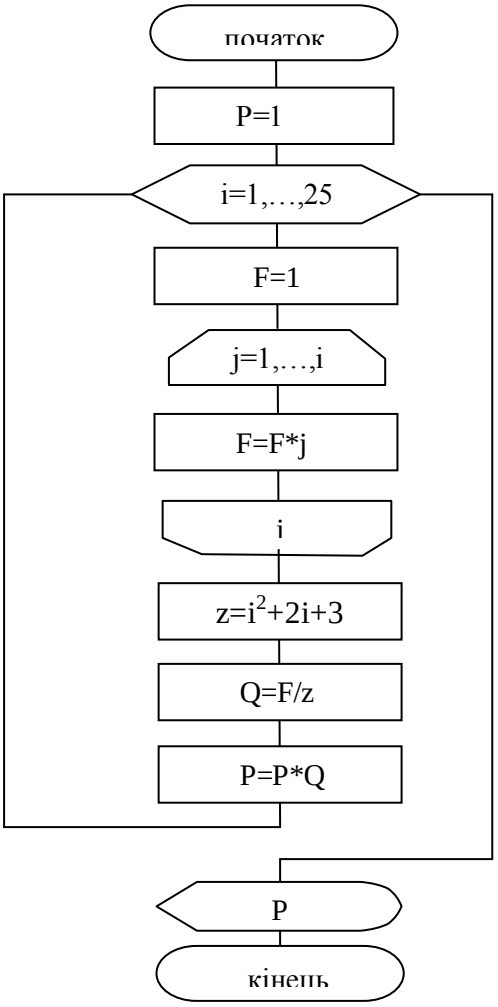
Формалізація задачі

Вхідних даних у програмі немає.

1. Оскільки необхідно отримати добуток, то треба підготувати змінну для цього: P=1. {2}
2. Нехай *i=1* (змінна циклу). {3}
3. Нехай *f=1* (змінна для підрахунку факторіалу). {4}
4. Обчислюємо факторіал *i!* : *f=f·i*. {5}
5. Обчислимо значення виразу:
   $Q=\frac{f}{i^2+2i+3}$. {6}
6. Обчислюємо P: *P=P*Q*. {7}
7. Якщо *i<25*, то {8}, інакше {9}.
8. *i = i+1*. {4}
9. Виводимо на екран P. {10}
10. Кінець програми.

Вихід: отримане значення добутку P.

Схема роботи програми



Лістинг програми

```
#include <conio.h>
#include <locale.h>
int main()
{
      setlocale(0,"");
      double P = 1.0;
      double factorial, z, Q;
      for (int i=1; i<=25; i++)
      {    // обраховуємо чисельник
```

```
            factorial = 1;
            int j=1;
            while (j<=i)
            {     factorial*=j;
                  j++;
            }
            z=(double)(i*i+2*i+3);
            Q = factorial/z;
           P=P*Q;
      }
      printf("\nРезультат: P=%e=%0.3f", P,P);
      _getch();
      return 0;
}
```

Результат виконання програми

```
Результат: P = 1,558555e+012 = 1558554764913,827
```

Порядок виконання роботи

1. Засвоїти теоретичний матеріал.
2. Розібратись з прикладами та процесом розробки наведених вище задачі.
3. Згідно з індивідуальним завданням розробити програми, що реалізують кожну із задач.
4. По кожній задачі підготувати звіт за формою, наведеною у лабораторній роботі №1 (титульний аркуш і висновки – загальні для усієї лабораторної роботи).

Варіанти індивідуальних завдань

Задача 1. Обчислити і вивести на екран у вигляді таблиці значення функції F на інтервалі від $X_{поч.}$ до $X_{кінц.}$ з кроком Δx. Значення *a, b, c,* $X_{поч.}$, $X_{кінц.}$, Δx вводити з клавіатури з обов'язковою перевіркою правильності введення.

1	$F=\begin{cases} ax^2+b & при \quad x<0 \quad і \quad b\neq 0 \\ \dfrac{x-a}{x-c} & при \quad x>10 \quad і \quad b=0 \\ \dfrac{x}{c} & в \quad інших \quad випадках \end{cases}$
2	$F=\begin{cases} \dfrac{1}{ax}-b & при \quad \lvert x+5\rvert<0 \quad і \quad c=0 \\ \dfrac{x-a}{x} & при \quad x>20 \quad і \quad c\neq 0 \\ \dfrac{10x}{c-4} & в \quad інших \quad випадках \end{cases}$

3	$F=\begin{cases} ax^2+bx+c & при \quad a<0 \quad і \quad c\neq 0 \\ \dfrac{-a}{x-c} & при \quad a>0 \quad і \quad b=0 \\ a(x-c) & в \quad інших \quad випадках \end{cases}$
4	$F=\begin{cases} -ax-c & при \quad x<0 \quad і \quad c\neq 0 \\ \dfrac{x-a}{-b} & при \quad x>0 \quad і \quad c=0 \\ \dfrac{bx}{c-a} & в \quad інших \quad випадках \end{cases}$
5	$F=\begin{cases} ax^2-b^2x & при \quad c<0 \quad і \quad b\neq 0 \\ \dfrac{x+a}{x+c} & при \quad c>0 \quad і \quad b=0 \\ x/c & в \quad інших \quad випадках \end{cases}$
6	$F=\begin{cases} -ax^2-b & при \quad x<5 \quad і \quad b\neq 0 \\ \dfrac{x-a}{x} & при \quad x>5 \quad і \quad c=0 \\ -b/c & в \quad інших \quad випадках \end{cases}$
7	$F=\begin{cases} -ax^2+b^2x & при \quad a<0 \quad і \quad b\neq 0 \\ x-\dfrac{a}{x-c} & при \quad a>0 \quad і \quad b=0 \\ 1+\dfrac{x}{c} & в \quad інших \quad випадках \end{cases}$
8	$F=\begin{cases} -ax^2+\dfrac{b}{c} & при \quad x<1 \quad і \quad b\neq 0 \\ \dfrac{x-a}{(x-c)^2} & при \quad x>1.5 \quad і \quad b=0 \\ \dfrac{x^2}{c^2} & в \quad інших \quad випадках \end{cases}$
9	$F=\begin{cases} ax^3+b^2+c & при \quad \lvert x\rvert<0.6 \quad і \quad b+c\neq 0 \\ \dfrac{x-a}{x-c} & при \quad b+c=0 \\ \dfrac{x}{c}-\dfrac{x}{a} & в \quad інших \quad випадках \end{cases}$
10	$F=\begin{cases} ax^3-b & при \quad x<0 \quad і \quad a\neq 0 \\ \dfrac{x-a}{x-c} & при \quad x>=10 \quad і \quad a=0 \\ \dfrac{x}{c}-\dfrac{c}{x} & в \quad інших \quad випадках \end{cases}$
11	$F=\begin{cases} a(x+c)^2-b & при \quad x<0 \quad і \quad b\neq 0 \\ \dfrac{x-a}{-c} & при \quad x>0 \quad і \quad b=0 \\ a+\dfrac{b}{c} & в \quad інших \quad випадках \end{cases}$

12	$F=\begin{cases} ax^2-cx+b & при \quad x+10<0 \quad і \quad b\neq 0 \\ \dfrac{x-a}{x-c} & при \quad x+10>0 \quad і \quad b=0 \\ \dfrac{-b}{a-c} & в \quad інших \quad випадках \end{cases}$
13	$F=\begin{cases} a(x+7)^2-b & при \quad x<5 \quad і \quad b\neq 0 \\ \dfrac{x-cd}{ax} & при \quad x>5 \quad і \quad b=0 \\ b/c & в \quad інших \quad випадках \end{cases}$
14	$F=\begin{cases} a-\dfrac{x}{10+b} & при \quad x<0 \quad і \quad b\neq 0 \\ \dfrac{x-a}{x-c} & при \quad x>0 \quad і \quad b=0 \\ 3a+b/c & в \quad інших \quad випадках \end{cases}$
15	$F=\begin{cases} ax^2-bx+c & при \quad x<3 \quad і \quad b\neq 0 \\ \dfrac{x-a}{x-c} & при \quad x>3 \quad і \quad b=0 \\ (a+b)/c & в \quad інших \quad випадках \end{cases}$

Задача 2. Використовуючи оператор вибору, розробити програму згідно з індивідуальним завданням.

1	Напишіть програму, яка запитує в користувача номер дня тижня, а потім виводить назву дня тижня чи повідомлення про помилку, якщо введені невірні дані (понеділок – 1, …, неділя – 7).
2	Напишіть програму, яка запитує у користувача число, а потім підносить його до степеню 2, 3, 4 або 5.
3	Написати програму, що підраховує кількість різних цифр у рядку символів, що вводиться з клавіатури.
4	Написати програму, що підраховує кількість різних символів «a, b, c, d, e, f» у рядку символів, що вводиться з клавіатури.
5	Напишіть програму, що підраховує пробіли, символи табуляції та нового рядка у вхідній послідовності символів, що вводяться з клавіатури.
6	Напишіть програму, що видаляє символ, який визначається користувачем, із вхідного потоку символів, що вводяться.
7	Напишіть програму, що перетворює літери, які вводяться з клавіатури, із нижнього регістра у верхній.
8	Для змінних **x** і **y** обрахувати вирази, в залежності від операції (+, -, *, /).
9	Написати програму, яка визначає, у якої фігури площа більша: коло, задане радіусом, чи квадрат, заданий стороною.
10	Напишіть програму, яка запитує в користувача номер місяця року, а потім виводить назву місяця або повідомлення про помилку, якщо введено невірні дані (січень – 1, …, грудень – 12).

11	Напишіть програму, яка запитує в користувача час , а потім виводить "AM", якщо введено час до полудня, або "PM", якшо час після полудня.
12	Напишіть програму, яка підраховує кількість голосних і приголосних літер у введеному слові або словосполученні.
13	Напишіть програму, яка запитує в користувача номер навчального урока, а у відповідь виводить час його початку і закінчення.
14	Напишіть програму, яка запитує в користувача цифру, а у відповідь виводить цю цифру словами.
15	Для змінної *x* обрахувати вираз в залежності від операції (*x*++, ++*x*, *x*--, --*x*).

Задача 3. Використовуючи оператори циклу, обчислити значення виразів, якщо *x* – дійсне, *n* – натуральне число. Числа *x* і *n* вводяться з клавіатури.

1	а) $\sum_{i=1}^{n}\left(\frac{1}{i!}+\sqrt{\lvert x\rvert}\right)$ б) $\prod_{i=1}^{52}\frac{i^2}{i^2+2i+3}$	6	а) $\sum_{k=1}^{n}\frac{1}{(k^2)!}$ б) $\prod_{i=2}^{n}\left(1-\frac{1}{i!}\right)^2$	11	а) $\sum_{i=1}^{n}\frac{x+\cos(ix)}{2^i}$ б) $\prod_{i=2}^{12}\frac{(i+1)!}{i+2}$
2	а) $\sum_{k=1}^{n}\frac{k}{k!}^{n}(-1)^{k+n}$ б) $\prod_{i=1}^{10}\left(\frac{1}{i!}-i^2\right)$	7	а) $\sum_{k=1}^{n}\frac{(-1)^{k+1}}{k\cdot(k+1)}$ б) $\prod_{i=2}^{n}\frac{(i-1)!}{i+2}$	12	а) $\sum_{k=1}^{n}k\cdot(k+1)\cdot\ldots\cdot k$ б) $\prod_{k=1}^{n}\left(\frac{k!}{k-1}-\sin^2(x)\right)$
3	а) $\frac{1}{n!}\sum_{i=1}^{n}(-1)^i\frac{(x+4)^i}{i!+3}$ б) $\prod_{k=1}^{n}\frac{k^3}{(k+1)^2}$	8	а) $\sum_{k=1}^{n}\frac{(2k)!}{(k^2)!}$ б) $\frac{1}{n!}\prod_{i=1}^{n}\frac{(i+1)^3}{i!-i^n}$	13	а) $\frac{1}{n!}\sum_{k=1}^{n}(-1)^k\frac{x^k}{(k!+1)}$ б) $\prod_{i=1}^{15}\frac{i+1}{i^2-i+1}$
4	а) $\sum_{i=1}^{n}\frac{x^i}{i!}$ б) $\prod_{k=1}^{n}\left(\frac{k}{k-1}-\sin(x^2)\right)$	9	а) $\sum_{k=1}^{n}\frac{k!}{(2k+1)^2}$ б) $\prod_{i=1}^{10}\left(2+\frac{1}{i!}\right)$	14	а) $\sum_{k=1}^{n}k^k x^{2k}$ б) $\prod_{i=1}^{10}\frac{i!}{i^2+2i+3}$
5	а) $\sum_{k=1}^{n}\frac{(-1)^k}{(2k+1)\cdot k!}$ б) $\prod_{k=2}^{n}\frac{x^2+k^2}{(k-1)^3}$	10	а) $\sum_{k=1}^{n}k^3\sum_{i=1}^{10}(k-i)^2$ б) $\prod_{i=2}^{10}\left(1-\frac{1}{i!}\right)^2$	15	а) $\sum_{i=2}^{n}i\cdot(i-1)^2$ б) $\prod_{i=2}^{n}\frac{(i-1)!}{i-2}$

***** ***Задача 3.*** Один з найпотужніших методів захисту інформації – це її шифрування. Розробити програму, яка для цілих $b, c \in [0; 9]$, що водяться з клавіатури у зашифрованому вигляді, обчислюватиме вираз та шифруватиме результат перед виведенням на екран:

$$a = \left(\sum_{i=1}^{10} \left(b \cdot i + 2 \cdot (c - i)^i \right) \right) \bmod 10$$

Варіант	Таблиця замін		Варіант	Таблиця замін		Варіант	Таблиця замін	
1	1 → 1	6 → 0	6	1 → 4	6 → 9	11	1 → 8	6 → 1
	2 → 3	7 → 8		2 → 6	7 → 5		2 → 7	7 → 6
	3 → 5	8 → 6		3 → 8	8 → 7		3 → 9	8 → 2
	4 → 7	9 → 4		4 → 2	9 → 3		4 → 0	9 → 3
	5 → 9	0 → 2		5 → 0	0 → 1		5 → 4	0 → 5
2	1 → 2	6 → 9	7	1 → 6	6 → 1	12	1 → 8	6 → 3
	2 → 4	7 → 7		2 → 7	7 → 2		2 → 1	7 → 5
	3 → 6	8 → 5		3 → 5	8 → 0		3 → 7	8 → 9
	4 → 8	9 → 3		4 → 4	9 → 3		4 → 2	9 → 0
	5 → 0	0 → 1		5 → 8	0 → 9		5 → 6	0 → 4
3	1 → 6	6 → 5	8	1 → 5	6 → 2	13	1 → 0	6 → 1
	2 → 7	7 → 4		2 → 4	7 → 8		2 → 9	7 → 4
	3 → 8	8 → 2		3 → 6	8 → 1		3 → 8	8 → 3
	4 → 9	9 → 3		4 → 3	9 → 0		4 → 6	9 → 2
	5 → 0	0 → 1		5 → 7	0 → 9		5 → 7	0 → 5
4	1 → 5	6 → 1	9	1 → 4	6 → 0	14	1 → 3	6 → 5
	2 → 4	7 → 3		2 → 3	7 → 9		2 → 6	7 → 4
	3 → 0	8 → 8		3 → 5	8 → 7		3 → 9	8 → 1
	4 → 9	9 → 7		4 → 6	9 → 8		4 → 7	9 → 2
	5 → 2	0 → 6		5 → 1	0 → 2		5 → 0	0 → 8
5	1 → 6	6 → 7	10	1 → 4	6 → 0	15	1 → 7	6 → 2
	2 → 8	7 → 5		2 → 5	7 → 9		2 → 4	7 → 9
	3 → 9	8 → 4		3 → 6	8 → 3		3 → 1	8 → 6
	4 → 0	9 → 3		4 → 8	9 → 2		4 → 8	9 → 0
	5 → 1	0 → 2		5 → 7	0 → 1		5 → 5	0 → 3

Контрольні питання

1. Які оператори відносять до умовних? Наведіть приклади використання умовних операторів.
2. За допомогою яких символів на схемах позначають умови?
3. Які оператори циклу ви знаєте? В чому між ними різниця?
4. Як перетворити оператор циклу з параметрами на оператор циклу з післяумовою?
5. За допомогою яких символів на схемах позначають циклічні процеси?

6. Як перетворити оператор циклу з передумовою на оператор циклу з передумовою?
7. Наведіть приклад обчислення будь-якого виразу за допомогою різних операторів циклу.
8. Чи можна в циклі *for* ініціалізувати відразу декілька змінних лічильників?
9. Чому слід уникати використання оператора *goto*?
10. Чи можна за допомогою оператора *for* організувати цикл, тіло якого не буде виконуватися?
11. Чи можна організувати цикл *while* всередині циклу *for*?
12. Як можна організувати нескінченні цикли? Наведіть декілька варіантів і поясніть їх.
13. Як можна вийти з нескінченних циклів?
14. Що відбувається при запуску нескінченного циклу?
15. Чи може оператор циклу не мати тіла? Чому?
16. Які оператори використовують для організації розгалуження у програмах?
17. Наведіть синтаксис і приклад оператору вибору. У яких випадках він використовується?
18. Чи можна віднести оператор вибору до умовних операторів? Чому?
19. Як перетворити оператор вибору в умовний оператор?
20. Для чого в операторах вибору використовують конструкцію *default*?
21. Для чого в операторах вибору необхідно використовувати *break*?
22. Наведіть приклад використання тернарного оператора "*?*".
23. Для чого служать оператори переривання *break* та *continue*? Наведіть приклад.

ЛАБОРАТОРНА РОБОТА №3

ГЕНЕРУВАННЯ ВИПАДКОВИХ ЧИСЕЛ. РОБОТА З ОДНОВИМІРНИМИ МАСИВАМИ

Мета роботи

- Дослідити функції генерування послідовностей псевдовипадкових чисел і навчитись їх застосовувати.
- Ознайомитись з можливостями створення одновимірних масивів і навчитись обробляти елементи масивів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Генерація псевдовипадкових чисел

В сучасних умовах для отримання випадкових чисел використовують різноманітні генератори, які поділяються на апаратні та програмні.

В апаратних генераторах джерелом випадкових чисел є шум в електронних приладах. Проте, використання апаратних генераторів вимагає наявності спеціального обладнання. В зв'язку з цим більш зручним вважається застосування програмних генераторів випадкових чисел.

Програмний генератор випадкових чисел являє собою програму, яка генерує послідовність чисел за деяким алгоритмом. Завдяки алгоритму така послідовність чисел цілком детермінована, тобто не може бути цілком випадковою. Її називають *послідовністю псевдовипадкових чисел*.

Для підключення вбудованого генератора псевдовипадкових послідовностей чисел в мові С використовують бібліотеку <*stdlib.h*>.

Функція ***rand():*** `int rand(void);`

Функція *rand()* генерує послідовність псевдовипадкових цілих чисел в діапазоні від 0 до *RAND_MAX*. Константа *RAND_MAX* визначена в бібліотеці <*stdlib.h*>. Його значення за замовчуванням може змінюватися залежно від реалізації, але воно не повинно бути меншим за 32767.

Функція ***srand():*** `void srand(unsigned int seed);`

Функція *srand()* використовується для ініціалізації генерування нової послідовності псевдовипадкових чисел, які отримуються за допомогою функції *rand()*. Якщо функція *srand()* викликається з одним і тим же початковим значенням, то послідовність псевдовипадкових чисел не зміниться. Тому в якості початкового заповнення зазвичай у програмах використовують системний час, який отримують за допомогою функції *time()* бібліотеки <*time.h*>.

```
long time(long *timeptr);
```

Функція *time* відповідно до системного годинника повертає кількість секунд, які пройшли з моменту часу 00:00:00 (значення часу за Гринвічем), тобто з 1 січня 1980 року. Якщо показчик *timeptr* не рівний NULL, то значення, яке повертається, також записується в *timeptr*.

Приклади програм для генерування випадкових чисел

Приклад №1. Програма генерує псевдовипадкове число від 1 до 10 і за допомогою клавіатури користувач намагається вигадати це число.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
int main ()
{ int iSecret, iGuess;
  srand (time(NULL)); /* ініціалізація початкового заповнення: */
  iSecret = rand() % 10 + 1; /* Генерація випадкового числа: */
  do {  printf ("Guess the number (1 to 10): ");
        scanf_s ("%d", &iGuess);
        if (iSecret < iGuess)
              puts("The secret number is lower");
        else
              if (iSecret>iGuess)
                    puts("The secret number is higher");
  } while (iSecret!=iGuess);
  puts ("\n\n\tCongratulations!");
  return 0;
}
```

Приклад №2. Програма генерує послідовності псевдовипадкових чисел (цілих і дійсних) на певних проміжках.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
int main ()
{ int n;
  printf ("\n\t\tГЕНЕРУВАННЯ ВИПАДКОВИХ ЧИСЕЛ \n\n\n");
  printf ("Введ1ть к1лькiсть чисел посл1довност1:  n = ");
  scanf_s ("%d", &n);
  srand (time(NULL) ); /* ініціалізація початкового заповнення: */
  printf("\n\n\tЦ1л1 випадков1 числа на проміжку  [-50,100]:\n");
  for (int i=0; i<n; i++)
  {     if (i%10==0) printf("\n");// Генерація цілого випадкового числа:
        int number = rand()%150-50;
        printf("%7d", number);
  }
  printf("\n\n\tД1йсн1 випадков1 числа на проміжку [-100,100]:\n");
  for (int i=0; i<n; i++)
  {     if (i%10 == 0) printf("\n");
        float number = (float)rand()/RAND_MAX*200 - 100.0;
        printf("%7.1f", number);
  }
  return 0;
}
```

Одновимірні масиви

Масив – це набір змінних одного типу, що мають одне і те ж ім'я. Доступ до конкретного елемента масиву здійснюється за допомогою індексу. У мові С всі масиви розташовуються в окремій безперервній області пам'яті. Перший елемент масиву розташовується з самою найменшою адресою, а останній – з найбільшою. Масиви можуть бути одновимірними і багатовимірними. Рядок – це масив символьних змінних, що закінчується спеціальним нульовим символом, це найбільш поширений тип масиву.

Загальна форма оголошення одновимірного масиву має вигляд:

```
тип ім'я_змінної [розмір];
```

Як і інші змінні, масив повинен бути оголошений явно, щоб компілятор виділив для нього певну область пам'яті (тобто розмістив масив). Тут *тип* позначає базовий тип масиву, що є типом кожного елемента. *Розмір* задає кількість елементів масиву. Наприклад, наступний оператор оголошує масив з 25 елементів типу *float* під ім'ям *account*:

```
float account [25];
```

Доступ до елемента масиву здійснюється за допомогою імені масиву й індексу, який поміщається в квадратних дужках після імені. Наприклад,

```
account [20] = 15.47;
```

Індекс першого елемента будь-якого масиву в мові С дорівнює нулю. Тому оператор

```
int num[11];
```

оголошує масив символів з 11 елементів – від num[0] до num [10].

Обсяг пам'яті, необхідний для зберігання масиву, безпосередньо визначається її типом і розміром. Для одновимірного масиву кількість байтів пам'яті обчислюється таким чином:

```
К-ть_байтів = sizeof (базовий_тип) × довжина_масиву.
```

Приклади програм з використанням одновимірних масивів

Приклад №3. Елементам масиву tod присвоюються значення від 0 до 55.

```
#include <stdio.h>
int main(void)
{
  int tod[55];
  int i;
  for(i=0; i<55; i++)
       tod[i] = i;
  for(i=0; i<55; i++)
       printf("%d ", tod[i]);
  return 0;
}
```

Приклад №4. Програма працює з трьома масивами, перший з яких ініціалізується цілими числами, другий заповнюються випадковими числами, а елементи третього – сумами відповідних елементів двох масивів.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
  const int n=15;
  int arr1[] = {1,3,-10,2,4,8,43,9,14,0,1,2,3,-5,-9};
  for (int i=0; i<n; i++)
        printf("%5d",arr1[i]);
  printf("\n");
  int arr2[n];
  for (int i=0; i<n; i++)
  {     arr2[i]=rand()%25+1;
        printf("%5d",arr2[i]);
  }
  printf("\n-----------------------------------------------------\n");
  int arr3[n];
  for (int i=0; i<n; i++)
  {     arr3[i]=arr1[i] + arr2[i];
        printf("%5d",arr3[i]);
  }
  printf("\n");
  _getch();
  return 0;
}
```

Приклад №5. Програма, що реалізує задачу з прикладу 4, але з використанням покажчиків.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
  const int n=15;
  int arr1[] = {1,3,-10,2,4,8,43,9,14,0,1,2,3,-5,-9};
  for (int i=0; i<n; i++)
        printf("%5d",arr1[i]);
  printf("\n");
  int arr2[n], arr3[n];
  for (int i=0; i<n; i++)
  {     arr2[i]=rand()%25+1;
        printf("%5d",arr2[i]);
  }
  printf("\n ---------------------------------------------\n");
  int *ptr1 = arr1;
  int *ptr2 = arr2;
  for (int i=0; i<n; i++)
  {     arr3[i]=*ptr1 + *ptr2;
        printf("%5d",arr3[i]);
        ptr1++;   ptr2++;
  }
  printf("\n");
  _getch();
  return 0;
}
```

Порядок виконання роботи

1. Засвоїти теоретичний матеріал. Розібратись з прикладами 1-5, відлагодити наведені в прикладах програми на комп'ютері.
2. Розробити власні програми, яка реалізують індивідуальне завдання.
3. Підготувати звіт по кожній задачі, який включатиме:
   - номер варіанту і текст індивідуального завдання;
   - формалізацію задачі;
   - схему роботи програми;
   - лістинг програми;
   - роздруківку трьох контрольних прикладів для кожної задачі;
   - висновки.

Варіанти індивідуальних завдань

Задача 1. Числа *m*, *n* і *k* *(3≤k≤10)* вводяться з клавіатури. Згенерувати і вивести на екран *m* цілих випадкових чисел з проміжку, вказаному у пункті а, та *n* дійсних чисел (виводити на екран з вказаною точностю) з проміжку, вказаному у пункті б. Виведення на екран здійснювати по *k* чисел у рядку.

1	а) [-25, 30]	б) [1, 5] з точністю до сотих
2	а) [13, 399]	б) [-2, 2] з точністю до десятих
3	а) [-200, 100]	б) [0, 10] з точністю до тисячних
4	а) [0, 125]	б) [3, 6] з точністю до сотих
5	а) [-100, 0]	б) [-1, 3] з точністю до десятих
6	а) [-45, 45]	б) [-5, 0] з точністю до тисячних
7	а) [-66, 666]	б) [0, 13] з точністю до сотих
8	а) [77, 127]	б) [-6, 10] з точністю до десятих
9	а) [-33, 333]	б) [1, 2] з точністю до тисячних
10	а) [-10, 10]	б) [-1, 0] з точністю до сотих
11	а) [-1000, 500]	б) [5, 10] з точністю до десятих
12	а) [-11, 111]	б) [2, 6] з точністю до тисячних
13	а) [13, 900]	б) [-1, 1] з точністю до сотих
14	а) [125, 500]	б) [10, 15] з точністю до десятих
15	а) [-444, 333]	б) [4, 8] з точністю до тисячних

Задача 2. Розробити программу, дотримуючись таких вимог:

- використовувати статичні масиви;
- число *n* (кількість елементів масиву) – іменована константа;
- елементи масиву – псевдовипадкові числа, згенеровані на інтервалі [*a*, *b*], де *a* і *b* вводяться з клавіатури (a<b);
- усі вхідні дані і також елементи масиву виводяться на екран.

1.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) суму від'ємних елементів масиву; 2) добуток елементів масиву, розташованих між максимальним і мінімальним елементами.
2.	В одновимірному масиві, що складається з *n* цілих елементів, обчислити: 1) суму додатних елементів масиву; 2) добуток елементів масиву, розташованих між максимальним за модулем і мінімальним за модулем елементами.
3.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) добуток елементів масиву з парними номерами; 2) суму елементів масиву, розташованих між першим і останнім нульовими елементами.
4.	В одновимірному масиві, що складається з *n* цілих елементів, обчислити: 1) суму елементів масиву з непарними номерами; 2) суму елементів масиву, розташованих між першим і останнім від'ємними елементами.
5.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) максимальний елемент масиву; 2) суму елементів масиву, розташованих до останнього додатного елемента.
6.	В одновимірному масиві, що складається з *n* цілих елементів, обчислити: 1) мінімальний елемент масиву; 2) суму елементів масиву, розташованих між першим і останнім додатними елементами.
7.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) номер максимального елемента масиву; 2) добуток елементів масиву, розташованих між першим і другим нульовими елементами.
8.	В одновимірному масиві, що складається з *n* цілих елементів, обчислити: 1) номер мінімального елемента масиву; 2) суму елементів масиву, розташованих між першим і другим від'ємними елементами.
9.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) максимальний за модулем елемент масиву; 2) суму елементів масиву, розташованих між першим і другим додатними елементами.
10.	В одновимірному масиві, що складається з *n* цілих елементів, обчислити: 1) мінімальний за модулем елемент масиву; 2) суму модулів елементів, розташованих після першого елемента, рівного нулю.

11.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) номер мінімального за модулем елемента масиву; 2) суму модулів елементів, розташованих після першого від'ємного елемента.
12.	В одновимірному масиві, що складається з *n* цілих елементів, обчислити: 1) номер максимального за модулем елементу масиву; 2) суму елементів масиву, розташованих після першого додатного елемента.
13.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) кількість елементів, що лежать в діапазоні від A до B (A і B – з клавіатури); 2) суму елементів масиву, розташованих після максимального елемента.
14.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) кількість елементів масиву, рівних 0; 2) суму елементів масиву, розташованих після мінімального елемента.
15.	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити: 1) кількість елементів масиву, більших за C (C вводиться з клавіатури); 2) добуток елементів масиву, розташованих після максимального за модулем елемента.

Задача 3. Змінити програму, що реалізує завдання 2, таким чином, щоб у програмі були використані покажчики. У звіт не включати формалізацію та схему програми.

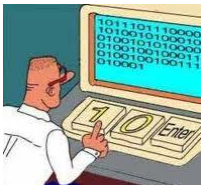
*** *Задача 3.*** В комп'ютерній системі зареєстровано таких користувачів: Alice, Bob, Carl з паролями Cooper, Dylan, Perkins відповідно. Розробити програму, яка визначатиме чи зможе користувач отримати доступ до читання (r), запису (w) або виконання файлів (x), на основі імені облікового запису користувача (так званий "логін") та пароля.

1	Alice Bob Carl	rx w rw	6	Alice Bob Carl	x r rw	11	Alice Bob Carl	rx rw x
2	Alice Bob Carl	rwx x w	7	Alice Bob Carl	r wx rw	12	Alice Bob Carl	w rwx rw
3	Alice Bob Carl	x wx r	8	Alice Bob Carl	rx wx rw	13	Alice Bob Carl	x x rw
4	Alice Bob Carl	w rw x	9	Alice Bob Carl	rw w rx	14	Alice Bob Carl	w rw rx
5	Alice Bob Carl	rwx r x	10	Alice Bob Carl	w wx r	15	Alice Bob Carl	r wx rw

Контрольні питання

1. Яким чином можна згенерувати випадкове число?
2. Для чого існує функція srand()?
3. Яким чином генеруються цілі випадкові числа на певному інтервалі?
4. Як згенерувати дійсні випадкові числа на певному інтервалі?
5. Що таке масиви? Як розташовуються елементи масивів у пам’яті?
6. Що собою являє рядок? В чому його особливість?
7. Які бувають масиви за розмірністю та способом виділення їм пам’ті?
8. Як оголошуюсь статичні масиви?
9. Яка роль ключового слова *const* при оголошенні розмірності масиву?
10. Що таке покажчик на змінну та адреса змінної? Що таке розіменування покажчиків?
11. Як звернутись до першого та останнього елементу масиву за допомогою покажчиків?
12. Яким чином обчислюється кількість байтів пам'яті, виділених масиву?
13. Наведіть фрагмент програми (за допомогою операторів *for*, *while* та *do-while*), що обчислює суму, добуток, кількість парних елементів одновимірного масиву, найбільше та найменше значення.

ЛАБОРАТОРНА РОБОТА №4



РОЗРОБКА ПРОГРАМ З ВИКОРИСТАННЯМ ДВОВИМІРНИХ МАСИВІВ

Мета роботи

- Засвоїти техніку використання двовимірних масивів при програмуванні мовою C/C++.
- Ознайомитись зі способами оголошення та ініціалізації двовимірних масивів.
- Засвоїти способи виділення пам’яті для динамічних масивів.
- Навчитись будувати математичну модель задачі.
- Засвоїти методи перевірки правильності вхідних даних.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Багатовимірні масиви

Двовимірний масив (матриця) можна представити як одновимірний масив, кожний елемент якого – масив. Тривимірний масив – це масив, кожний елемент якого являє собою двовимірну матрицю.

```
char Matrix2D[6][9];              // Двовимірний масив 6x9 елементів
unsigned long Arr3D[4][2][8];     // Тривимірний
```

Багатовимірні масиви ініціалізуються в порядку якнайшвидшої зміни самого правого індексу: спочатку відбувається присвоєння початкових значень всіх елементів останнього індексу, потім попереднього і т. д.:

```
int Mass[3][2][4] = {1,2,3,4,5,6,7,8,9,10,11,12,
                     13,14,15,16,17,18,19,20,21,22,23,24};
```

Рекомендується для наочності групувати дані за допомогою проміжних фігурних дужок:

```
int Mass[3][2][4]={{{1,2,3,4},{5,6,7,8}},
                   {{9,10,11,12},{13,14,15,16}},
                   {{17,18,19,20},{21,22,23,24}};
```

Для багатовимірних масивів при ініціалізації дозволяється опускати лише величину першої розмірності:

```
int main()
{
  char x[][3]={{9,8,7},{6,5,4},{3,2,1}};
  for(int i=0; i<3; i++)
  {
        for(int j=0; j<3; j++)
              printf("%d ", (int)x[i][j]);
        printf("\n");
  }
  return 0;
}
```

Динамічне виділення пам'яті під масиви

Змінні у програмах повинні розміщатися в одному з трьох місць: в області даних програми, в області стеку, в області вільної пам'яті (купи).

Кожній змінній у програмі може відводитися пам'ять або статично (в момент завантаження), або динамічно (у процесі виконання програми).

До цих пір всі використовувані масиви оголошувались статично, а отже, зберігали значення своїх елементів в області даних. Якщо кількість елементів невелика, таке розміщення виправдано. Але досить часто виникають випадки, коли необхідно мати великі масиви даних або розмір масиву заздалегідь не може бути визначений. Тут на допомогу приходить можливість використання динамічної пам'яті.

Для того, щоб у пам'яті можна було розмістити будь-який динамічний об'єкт, для нього необхідно попередньо виділити відповідне місце. По завершенні роботи з об'єктом виділену пам'ять необхідно звільнити.

Виділення пам'яті можна здійснювати двома способами.

I спосіб: функції ***malloc(), calloc()***, ***free()***. Ці функції описані у заголовочному файлі <*stdlib.h*> або <*malloc.h*>

Функція *malloc(size)* виділяє *size* байтів з купи. У випадку успішного виділення пам'яті покажчик встановлюється на виділений блок пам'яті. При невдалому виділенні пам'яті функція повертає *NULL*.

Функція *calloc(num, size)*, окрім виділення області пам'яті під масив об'єктів, ще здійснює ініціалізацію елементів масиву нульовими значеннями. Тут *num* вказує, скільки елементів буде зберігатися у масиві, а *size* – розмір кожного елемента у байтах.

Наприклад, якщо необхідно виділити пам'ять для масиву з *n* цілих чисел типу long, це можна зробити за допомогою оператора:

```
long * fptr = (long *)malloc(n*sizeof(long));
```

Якщо необхідно виділити пам'ять під двовимірний масив розмірності $m \times n$, це можна зробити за допомогою оператора

```
float* fptr = (float*)malloc(n*m*sizeof(float));
```

Функція *free*(**block*) звільняє пам'ять, на яку вказує покажчик *block*.

II спосіб: функції ***new(), delete().*** Ці функції з'явилися в C++.

malloc(), calloc(), free() працюють і в C, і в C++. Нові оператори гнучкого розподілення пам'яті *new()* і *delete()* мають додаткові можливості.

Якщо оператори *malloc(), calloc()* повертають пустий покажчик, який далі перетворюється до потрібного типу, то оператор *new()* повертає покажчик на той тип, для якого виділяється пам'ять, і додаткових перетворень не потребує.

Синтаксис операторів:

```
тип *ім'я_масиву = new тип [тип число];
. . .
delete[] ім'я;
```

Приклад використання двовимірного статичного масиву

Задача №1. Реалізувати програму, яка обчислює суму елементів, що розташовані на обох діагоналях квадратної матриці.

Формалізація задачі

1. Вхідні дані: N – максимальна розмірність матриці (матриця квадратна, оскільки мова йде про діагоналі матриці; n – реальна розмірність матриці A.
2. Вихідні дані: S – сума елементів обох діагоналей.
3. Здійснити перевірку коректності вхідних даних:
4. Для заповнення елементів матриці використаємо генератор випадкових чисел.
5. Для можливості здійснення перевірки правильності обчислень доцільно вивести матрицю на екран.

Математична модель задачі.

Нехай задана квадратна матриця A розміром $n \times n$:

$$A = \begin{array}{|c|c|c|c|c|} \hline A_{00} & A_{01} & A_{02} & \dots & A_{0,n-1} \\ \hline A_{10} & A_{11} & A_{12} & \dots & A_{1,n-1} \\ \hline A_{20} & A_{21} & A_{22} & \dots & A_{2,n-1} \\ \hline \dots & \dots & \dots & \dots & \dots \\ \hline A_{n-1,0} & A_{n-1,1} & A_{n-1,2} & \dots & A_{n-1,n-1} \\ \hline \end{array}$$

Необхідно знайти

$$S = S_1 + S_2,$$

де

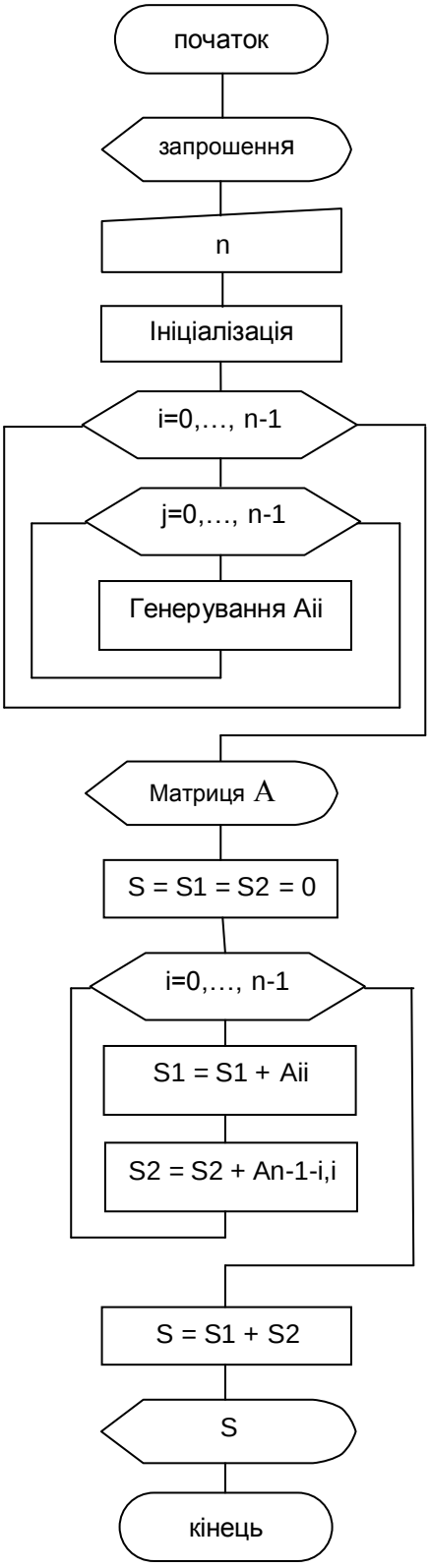
$$S_1 = A_{00} + A_{11} + \dots + A_{n-1,n-1} = \sum_{i=0}^{n-1} A_{ii};$$

$$S_2 = A_{n-1,0} + A_{n-2,1} + \dots + A_{0,n-1} = \sum_{i=0}^{n-1} A_{n-1-i,,i};$$

Для знаходження S_1 і S_2 доцільно використати оператор циклу *for*.

Оскільки для обчислення обох частинних сум S_1 і S_2 змінна циклу *i* змінюється однаково *(i=0,…,n-1)*, то обрахування обох сум можна виконати в одному циклі.

Схема роботи програми



Лістинг програми

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
#include <locale.h>
#define N 10
using namespace std;
int main()
{ setlocale(0,"");
  int a[N][N];
  int n=N+1;
  int A=0, B=0, S, S1=0, S2=0;
  while(n>N) // перевірка правильності введення
  {
    printf("\nВведіть розмірність матриці: n = ");
    scanf_s("%d",&n);
  }
  while (A>=B)
  {
     printf("\nВведіть границі проміжку A і B: ");
     scanf_s("%d %d",&A, &B);
  }
  printf("\n\nМатриця A:");
  for (int i=0;i<n;i++)
  {    printf("\n\n");
       for (int j=0;j<n;j++)
       {    a[i][j]=rand()%(B-A)+A;
            printf("%5d",a[i][j]);
       }
  }
  for (int i=0;i<n;i++)// підрахунок суми
  {
       S1 += a[i][i];
       S2 += a[n-1-i][i];
  }
  S = S1 + S2;
  printf("\n\nСума діагональних елементів S = %7d",S);
  return 0;
}
```

Результати виконання програми:

```
Введіть розмірність матриці: n = 5
Введіть границі проміжку A і B: 1 10

Матриця A:
    6    9    8    5    9
    2    4    1    8    3
    9    3    8    7    8
    6    8    9    4    1
    1    7    6    1    5
Сума д?агональних елементів S =      61
```

Приклад використання динамічного двовимірного масиву

Задача №2. Заповнити прямокутну матрицю випадковими числами з інтервалу (a, b). Створити матрицю, що містить стільки ж стовпців, як і вхідна матриця, і 3 рядки: 1-й рядок – максимальні значення відповідних стовпців, 2-й – середні арифметичні значення елементів стовпця, 3-й рядок – мінімальні значення відповідних стовпців.

Математична модель задачі.

Нехай маємо матрицю A розмірністю n×m.

A =

A_{00}	A_{01}	A_{02}	…	$A_{0,m-1}$
A_{10}	A_{11}	A_{12}	…	$A_{1,m-1}$
A_{20}	A_{21}	A_{22}	…	$A_{2,m-1}$
…	…	…	…	…
$A_{n-1,0}$	$A_{n-1,1}$	$A_{n-1,2}$	…	$A_{n-1,m-1}$

Необхідно отримати матрицю B:

B =

B_{00}	B_{01}	B_{02}	…	$B_{0,m-1}$
B_{10}	B_{11}	B_{12}	…	$B_{1,m-1}$
B_{20}	B_{21}	B_{22}	…	$B_{2,m-1}$

$B_{0j} = \max\{A_{ij},\ i=0, \ldots, n-1\}$ – максимальне значення серед елементів j-ого стовпця (j=0,…,m-1);

$B_{1j} = \text{avg}\{A_{ij},\ i=0, \ldots, n-1\} = (A_{0j}+A_{1j}+\ldots+A_{n-1,j})/n$ – середнє значення елементів стовпця (j=0,…,m-1);

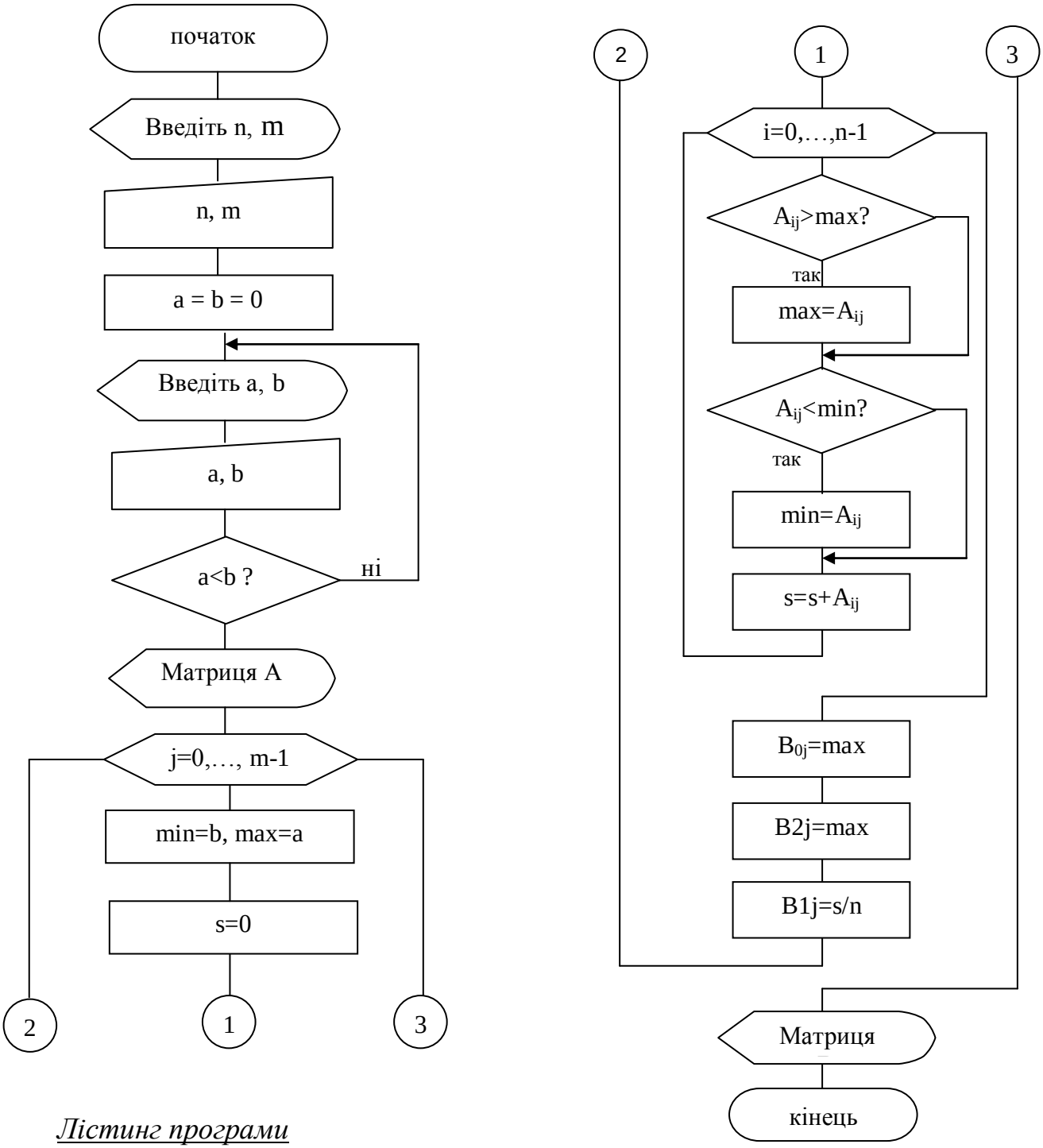
$B_{2j} = \min\{A_{ij},\ i=0, \ldots, n-1\}$ – максимальне значення серед елементів стовпця (j=0,…,m-1).

Формалізація завдання

1. Вхідні дані: n, m – кількість рядків і стовпців матриці A відповідно, a, b – границі інтервалу для елементів матриці.
2. Вихідні дані: матриця B, що містить максимальні, середні, мінімальні значення стовпців.
3. Розмірність матриці задається у процесі роботи програми, отже, для неї пам’ять будемо виділяти динамічно (з купи).
4. Границі інтервалу (a, b) вводимо з клавіатури, елементи матриці заповнюємо за допомогою генератора випадкових чисел.
5. Матрицю B доцільно оголосити дійсною, оскільки серед її елементів будуть значення середнього арифметичного, а це – результат розрахунку.Пошук мінімального і максимального елементів будемо здійснювати одночасно, перебираючи і порівнюючи усі елементи стовпця по черзі (використаємо оператор циклу).

6. Для знаходження середнього арифметичного значення треба накопичувати суму елементів стовпця, що також можна робити у тому ж самому циклі.

Схема роботи програми



Лістинг програми

```
#include <iostream>
#include <conio.h>
#include <new.h>
#include <locale.h>
```

```
using namespace std;

int main()
{  setlocale(0,"");
   int n, m, a=0, b=0;
   cout<<" \nВвед1ть розм1рності масиву n і m:\n";
   cin>>n>>m;
   cout<<"\n n="<<n<<"  m="<<m;
   while (a>=b)
   {        //перевірка правильності
      cout<<" \n\n  Введ1ть границ1 1нтервалу a і b:\n";
      cin>>a>>b;
      cout<<"\n a="<<a<<" b="<<b;
    }
    int** arr = new int*[n];
    cout<<"\n\n  Початковий  масив:\n";
    for (int i=0;i<n;i++)
    { arr[i] = new int[m];
      cout<<"\n";
      for (int j=0;j<m;j++)
      {
            arr[i][j]=rand()%(b-a)+a;
            printf("%7d",arr[i][j]);
      }
    }
    float** brr = new float*[3];
    for (int i=0;i<3;i++)
        brr[i] = new float[m];
    for (int j=0;j<m;j++)
    {  // по стовпцях
        int min=b;// початкове значення мінімуму у стовпці
            int max=a;  // початкове значення максимуму  у стовпці
            float s=0;  // для накопичення суми у сер. арифм.  у стовпці
            for (int i=0;i<n;i++)
            {
                  if (arr[i][j]>max)  max = arr[i][j];
                  if (arr[i][j]<min)  min = arr[i][j];
                  s += arr[i][j];
            }
            s = s/n;     // обчислюємо сер.арифм.
            brr[0][j] = max; brr[1][j] = s;      brr[2][j] = min;
      }
      cout<<"\n\n  Результуючий масив:\n";
      for (int i=0;i<3;i++)
      {
            cout<<"\n";
            for (int j=0;j<m;j++)
                  if (i==1) printf("%7.2f",brr[i][j]);
                  else      printf("%7.0f",brr[i][j]);
      }
      delete[] arr;      delete[] brr;

      _getch();
      return 0;
}
```

Результат роботи програми

```
 Введіть розмірності масиву n і m:
5
10

 n = 5    m = 10

 Введіть границі інтервалу a і b:
-50
100

 a = -50    b = 100

 Початковий  масив:

   -9    -33    -16     50     69     74     28     58     62    -36
  -45     45    -19    -23     11     -9     95     42    -23    -14
   91      4    -48    -47     92     32    -29     66     18     45
   -3     76     21     88     19     62    -33     -1     35     94
    3     61     72    -17     73    -36     91     11      3     68

 Результуючий масив:

   91     76     72     88     92     74     95     66     62     94
 7,40  30,60   2,00  10,20  52,80  24,60  30,40  35,20  19,00  31,40
  -45    -33    -48    -47     11    -36    -33     -1    -23    -36
```

Рекомендації

1. Динамічне виділення пам’яті для одновимірного масиву *A* (цілих чисел) розміром *n* і заповнення елементів масиву випадковими числами:

```
int* A = new int[n];
for (int i=0;j<n;j++)
     A[i]=rand()%RAND_MAX+1;
```

2. Динамічне виділення пам’яті для двовиимірного масиву *arr* (цілих чисел) розміром *n*×*m* та його заповнення випадковими числами:

```
int** A = new int*[n];
for (int i=0;i<n;i++)
{
     arr[i] = new int[m];
     for (int j=0;j<m;j++)
          arr[i][j]=rand()%RAND_MAX+1;
}
```

Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом.
2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програму на своєму комп’ютері.
3. Розробити власні програми, які реалізує індивідуальне завдання.
4. Підготувати звіт, який включатиме:
   - варіант і текст індивідуального завдання;
   - формалізацію завдання (включаючи математичну модель задачі);
   - схему роботи програми;
   - лістинг програми;
   - роздруківку трьох контрольних прикладів (вигляд екрану з результатами (різні значення вхідних даних);
   - висновки.

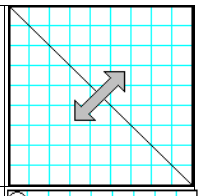
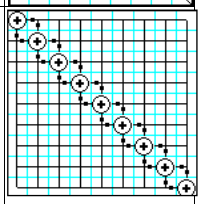
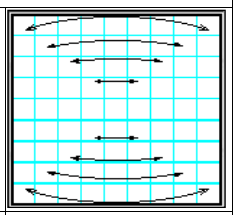
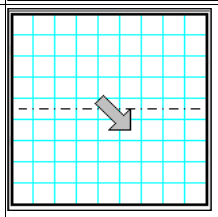
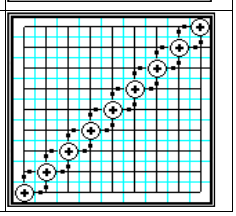
Варіанти індивідуальних завдань

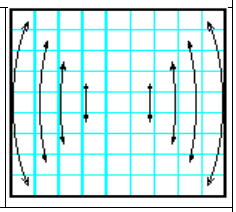
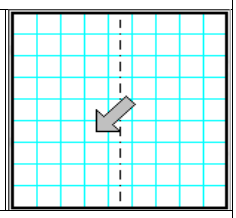
Задача 1. Розробити програму, дотримаючись таких вимог: використовувати статичні масиви; максимальні розміри масиву (N *і* M) – статичні константи; реальні розміри масиву n *і* m ($n<N$, $m<M$) – ввести з клавіатури (при цьому здійснювати перевірку правильності введення даних); елементи масиву – псевдовипадкові числа, згенеровані на інтервалі $[a, b]$, де a і b ($a<b$) вводяться з клавіатури; усі вхідні дані і також елементи масиву виводити на екран.

1	Реалізувати програму, яка міняє місцями перший і останній стовпці квадратної матриці.
2	Реалізувати програму, яка додає перший і останній рядки квадратного масиву і записує результат у останній стовпець.
3	Реалізувати програму, яка міняє значення елементів квадратної матриці ні значення відповідних елементів заданого одновимірного масиву.
4	Реалізувати програму, яка додає відповідні елементи двох заданих масивів і заносить результат у третій масив. Усі три масиви мають однакові розмірності $(n \times m)$.
5	Реалізувати програму, яка міняє місцями перший рядок і останній стовпець квадратної матриці.
6	Реалізувати програму, яка міняє місцями діагоналі квадратної матриці.
7	Реалізувати програму, яка сумує елементи рядків двовимірного масиву і заносить результат в одновимірний масив, розмірність якого дорівнює числу рядків двовимірного масиву.
8	Реалізувати програму, яка знаходить максимальний за модулем елемент заданого двовимірного масиву.
9	Реалізувати програму, яка міняє місцями останній рядок і перший стовпець квадратної матриці.
10	Реалізувати програму, яка додає перший і останній стовпці квадратної матриці і записує результат на місце першого рядка.
11	Реалізувати програму, яка міняє елементи заданого стовпця на значення відповідних елементів одновимірного масиву.
12	Реалізувати програму, яка перемножує відповідні елементи двох заданих масивів і заносить результат у третій масив. Розмірності усіх масивів однакові.
13	Реалізувати програму, яка міняє місцями останній рядок і перший стовпець квадратної матриці.
14	Реалізувати програму, яка сумує елементи стовпців двовимірного масиву і зносить результат в одновимірний масив, розмірність якого дорівнює числу стовпців двовимірного масиву.
15	Реалізувати програму, яка знаходить номер рядка заданого двовимірного масиву, що має максимальну за модулем суму елементів.

Задача 2. Змінити код програми (задача №1) таким чином, щоб використовувались не статичні масиви і статичні константи для їх оголошення, а динамічні масиви. У звіт включити лише лістинг програми та роздруківку трьох контрольних прикладів.

Задача 3. Розробити програму, дотримаючись таких вимог: використовувати динамічні масиви; реальні розміри масиву *n* і *m* ввести з клавіатури; елементи масиву – псевдовипадкові числа на інтервалі [*a*,*b*], де *a* і *b* (*a*<*b*) вводяться з клавіатури; усі вхідні дані, елементи вхідного і вихідного масивів виводяться на екран у зручному для перегляду вигляді. Підготувати звіт у повному обсязі.

1	Заповнити матрицю А випадковими числами. Відобразити матрицю симетрично відносно головної діагоналі. Знайти максимальний і мінімальний елемент головної діагоналі.	
2	Заповнити матрицю А випадковими числами. Створити одновимірний масив В, елементи якого – кількість додатних чисел відповідного рядка. На головній діагоналі розмістити суми елементів, які лежать на тому ж рядку і у тому ж стовпці.	
3	Дана цілочисельна прямокутна матриця А. Визначити кількість рядків, що не містять жодного нульового елемента. Створити одновимірний масив В, елементами якого є максимальні значення відповідних рядків матриці А.	
4	Дана цілочисельна прямокутна матриця А. Ввести ціле число *c (a*<*c*<*b)*. Підрахувати середнє арифметичне значення елементів головної і побічної діагоналей. Створити іншу матрицю В, в якій елементи, менші за *c* замінити на 0, а елементи, більші за *c*, замінити на 1.	
5	Заповніть матрицю А випадковими числами. Знайти середнє арифметичне усіх елементів матриці. Відобразити головну і побічну діагоналі симетрично щодо вертикальної осі.	
6	Заповнити прямокутну матрицю випадковими числами. Відобразити верхню половину матриці на нижню дзеркально симетрично відносно горизонтальної осі. Знайти і вивести на друк послідовність елементів матриці, які попадають в інтервал [a/2, b/2].	
7	Заповнити матрицю випадковими числами. Знайти номери рядків і стовпців, які містять мінімальний і максимальний елементи матриці. На побічної діагоналі розмістити суми елементів, які лежать на тому ж рядку і у тому самому стовпці.	

8	Заповнити матрицю А випадковими дійсними числами. Створити матрицю В, яка має стільки ж рядків, як і матриця А, і 5 стовпчиків: 1-й стовпчик буде містити мінімальні елементи відповідного рядка, 2-й – максимальні, 3-й – середні арифметичні значення, 4-й – кількість додатних елементів, 5-й – кількість від'ємних елементів відповідного рядка.
9	Заповнити матрицю випадковими числами. Відобразити головну і побічну діагоналі симетрично відносно горизонтальної осі. Знайти максимальне і мінімальне значення головної і побічної діагоналей. 
10	Дана прямокутна цілочисельна матриця. Визначити добуток елементів в тих рядках, які не містять від'ємних елементів. Підрахувати максимум і мінімум серед сум елементів діагоналей на головній і побічній діагоналях матриці.
11	Заповнити прямокутну матрицю випадковими числами. Відобразити праву половину матриці на ліву дзеркально симетрично щодо вертикальної осі. Підрахувати кількість додатних і від'ємних елементів у матриці. 
12	Дана прямокутна дійсна матриця. Визначити суму елементів в тих стовпцях, які не містять від'ємних елементів. Визначити омер рядка матриці, сума елементів якої максимальна.
13	Дана прямокутна дійсна матриця. Підрахувати кількість парних і непарних елементів на головній і побічній діагоналях матриці. Визначити номер стовпця, сума елементів якого мінімальна.
14	Дана прямокутна цілочисельна матриця. Знайти середнє арифметичне усіх елементів матриці. Знайти і вивести на друк послідовність елементів матриці, які попадають в інтервал [a/4, b/4].
15	Дана цілочисельна матриця А. Ввести ціле число *c (a<c<b)*. Створити іншу матрицю В, в якій елементи, більші за *c* замінити на 1, а елементи, більші за *c*, замінити на -1. Підрахувати середнє арифметичне значення елементів головної і побічної діагоналі.

***** ***Задача 4.*** Одна із задач операційних систем – розподіл пам'яті між прикладними програмами та між користувачами. Написати модуль для операційної системи, який не дозволятиме користувачам Alice, Bob та Carl отримувати оперативної пам'яті більше, ніж їм дозволено.

При цьому передбачити, що потреби користувачів можуть змінюватись протягом часу – вони можуть надавати запити до операційної системи щодо виділення та звільнення ділянок пам'яті залежно від своїх потреб, на які повинен реагувати даний модуль.

1	Alice Bob Carl	136 54980 900	6	Alice Bob Carl	15002 41900 385	11	Alice Bob Carl	153 7833 33000
2	Alice Bob Carl	20005 5304 80	7	Alice Bob Carl	490 46800 3160	12	Alice Bob Carl	17005 7649 503
3	Alice Bob Carl	4500 128 64000	8	Alice Bob Carl	4624 314 60001	13	Alice Bob Carl	5028 416 19003
4	Alice Bob Carl	4300 1000 941	9	Alice Bob Carl	56123 804 2500	14	Alice Bob Carl	45612 800 176
5	Alice Bob Carl	542 9800 12000	10	Alice Bob Carl	7800 59001 6824	15	Alice Bob Carl	5001 49025 230

Контрольні питання

1. В чому різниця між статичними та динамічними масивами?
2. Коли і звідки видялється пам’ять для статичних і динамічних масивів?
3. Як оголосити одновимірний динамічний масив?
4. Як оголосити динамічний двовимірний масив?
5. Наведіть фрагмент коду для заповнення масиву цілими (дійсними) випадковими числами.
6. Наведіть фрагмент коду для обчислення суми (добутку) елементів одновимірного масиву?
7. Наведіть фрагмент коду для підрахунку суми (добутку) елементів певного рядка (стовпця) двовимірного масиву?

ЛАБОРАТОРНА РОБОТА №5

РОБОТА З ТЕКСТОВИМИ РЯДКАМИ

Мета роботи

- Вивчити можливості оголошення та ініціалізації рядків.
- Навчитись вводити рядки з клавіатури і виводити їх на дисплей.
- Дослідити основні функції роботи з рядками та реалізовувати найпростіші операції з текстовими рядками.
- Навчитись використовувати основні функції бібліотек мови С для роботи з рядками (масивами символів).

ТЕОРЕТИЧНІ ВІДОМОСТІ

Основні функції роботи з символами та рядками

Стандартна бібліотека мови С володіє багатим і різноманітним набором функцій для обробки символів (табл. 5.1) і рядків (табл. 5.2).

Таблиця 5.1 – Функції для роботи із символами (<ctype.h>)

Прототип функції	Опис
`int` **`isalnum`**`(int c);`	Перевірка, чи є символ літерою або цифрою
`int` **`isalpha`**`(int c);`	Перевірка, чи є символ літерою
`int` **`iscntrl`**`(int c);`	Перевірка, чи є символ керуючим
`int` **`isdigit`**`(int c);`	Перевірка, чи є символ десятковою цифрою
`int` **`isgraph`**`(int c);`	Перевірка, чи є символ видимим
`int` **`isprint`**`(int c);`	Перевірка, чи є символ видимим, включаючи пробіл
`int` **`islower`**`(int c);`	Перевірка, чи є символ літерою нижнього регістру
`int` **`ispunct`**`(int c);`	Перевірка, чи є символ знаком пунктуації
`int` **`isspace`**`(int c);`	Перевірка, чи є символ пробільним
`int` **`isupper`**`(int c);`	Перевірка, чи є символ літерою верхнього регістру
`int` **`isxdigit`**`(int c);`	Перевірка, чи є символ шістнадцятковою цифрою
`int` **`tolower`**`(int c);`	Перетворення символу в нижній регістр
`int` **`toupper`**`(int c);`	Перетворення символу у верхній регістр

Таблиця 5.2 – Функції бібліотеки введення/виведення (<*stdio.h*>)

Прототип функції	Опис
`int` **`getchar`**`(void)`	Вводить наступний символ зі стандартного пристрою введення і повертає його в форматі цілого.
`char *`**`gets`**`(char *s)`	Вводить символи зі стандартного пристрою введення в масив s до тих пір, поки не зустріне символ кінця рядка або індикатор кінця файлу.
`int` **`putchar`**`(int c)`	Виводить символ, який зберігається в с.

int **puts**(const char *s)	Виводить рядок s та перехід на наступний рядок
int **sprintf**(char *buf, const char *format [,arg1, …])	Виконує форматоване виведення у рядок buf. Параметр format задає спосіб відображення значень змінних arg1, …. Дія функції sprintf аналогічна дії функції printf, але виведення виконується в рядок-буфер, а не на екран.
int **sscanf**(const char *s, const char *format [, address1, …])	Виконує дії, еквівалентні scanf за винятком того, що введення здійснюється з масиву s, а не з клавіатури.

Рядкові функції працюють з масивами символів (рядками), що закінчуються символом кінця рядка. У мові С для роботи з рядковими функціями використовується заголовок <*string.h*>. У ньому визначено тип *size_t* – тип результату, який утворюється після застосування оператора *sizeof()* і являє собою різновид цілого без знака. У таблиці 5.3 наведено найбільш популярні і підтримувані більшістю компіляторів функції.

Таблиця 5.3 – Функції роботи з рядками (<*string.h*>)

Прототип функції	Опис
char *__strcpy__(char *__s1__, const char *__s2__)	Копіює рядок s2 в масив s1. Повертає значення s1.
char *__strncpy__(char *__s1__, const char *__s2__, size_t __n__)	Копіює не більше, ніж n символів рядка s2 в масив s1. Повертає значення s1.
char *__strcat__(char *__s1__, const char *__s2__)	Об'єднує рядок s2 з рядком масива s1. Перший символ рядка s2 переписує символ NULL рядка s1. Повертає значення s1.
char *__strncat__(char *__s1__, const char *__s2__, size_t __n__)	Об'єднує не більше, ніж n символів рядка s2 з рядком s1. Перший символ рядка s2 переписує символ NULL рядка s1. Повертає значення s1.
int __strcmp__(const char *__s1__, const char *__s2__)	Порівнює рядок s1 з рядком s2. Функція повертає 0, значення менше 0 або більше 0, якщо s1 рівна, менше або більше, ніж s2.
int __strncmp__(const char *__s1__, const char *__s2__, size_t __n__)	Порівнює n символів рядків s1 і s2. Функція повертає 0, значення, менше 0 або більше 0, якщо s1 відповідно рівний, менший або більший, ніж s2.
char *__strchr__(const char *__s__, int __c__)	Знаходить позицію першого входження символа с в рядок s. Якщо с знайдено, функція повертає покажчик на с в рядку s. Інакше повертається покажчик NULL.
char *__strrchr__(const char *__s__, int __c__)	Знаходить позицію останнього входження символу с в рядок s. Якщо с знайдено, то повертається покажчик на с в рядку s. Інакше повертається покажчик NULL.
size_t __strcspn__(const char *__s1__, const char *__s2__)	Повертає довжину початкового сегмента рядка s1, що містить тільки ті символи, що не входять в s2.
size_t __strspn__(const char *s1, const char *__s2__)	Визначає і повертає довжину початкового сегмента рядка s1, що містить тільки ті символи, що входять в s2.

char ***strstr***(const char ***s1,*** const char ***s2***)	Знаходить позицію першого входження рядка s2 в рядок s1. Якщо знайдено, повертається покажчик підрядка в рядку s1. Інакше повертається покажчик NULL.
char ***strtok*** (const char ***s1,*** const char ***s2***)	Послідовні виклики функції виконують розбиття рядка s1 на лексеми (слова), розділені символами, які містяться в s2. При першому виклику функція отримує в якості аргументу рядок s1, а при наступних викликах, щоб продовжити розбиття того ж рядка, першим аргументом передається NULL. При кожному виклику повертається покажчик на поточну лексему рядка s1. Якщо лексем в рядку не залишилось, то повертається NULL.
size_t ***strlen***(const char ***s***)	Визначає довжину рядка s (до символу NULL).

Оскільки в С не передбачений автоматичний контроль порушення меж масивів, вся відповідальність за їх переповнення лягає на програміста.

Приклад розробки програми для роботи з рядками і символами

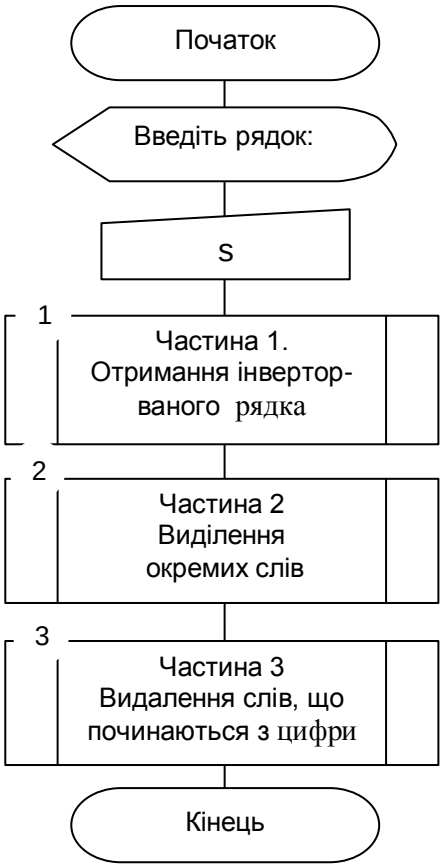
Задача. З клавіатури вводиться текстовий рядок. Скласти програму, яка: а) інвертує рядок, подаючи його у зворотному вигляді; б) підраховує кількість чисел у тексті; в) видаляє всі слова, що починаються з цифри.

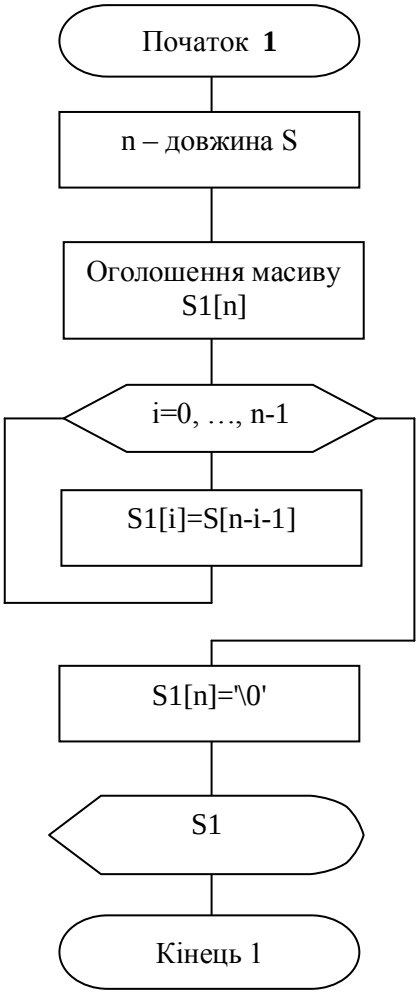
Формалізація задачі

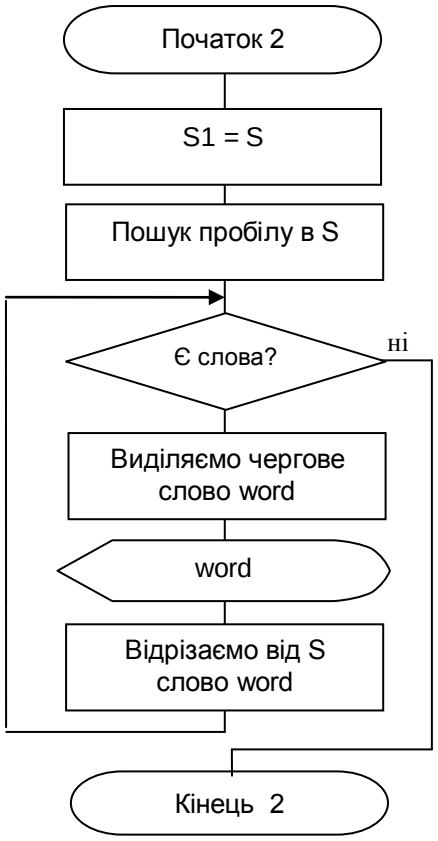
1. Вхідний рядок S представимо у вигляді масиву символів.
2. Підрахуємо n – кількість символів у рядку.
3. Далі програму розділимо на три послідовних частини.
   a) Отримання інвертованого рядка:
   – підготуємо рядок S1 такого самого розміру n;
   – здійснимо в циклі (i=0, …, n-1) присвоєння:
   – $S1_i = S_{n-i-1}$. (тут враховуємо те, що останній символ рядка S дорівнює '\0');
   – додаємо в массив S1 символ закінчення (щоб обірвати рядок).
   b) Виділення окремих слів.
   – запам’ятовуємо S в S1 (щоб не зіпсувати початковий рядок);
   – переглядаємо рядок S по словах (виділяємо частину рядка до пробілу). Це доцільно зробити в циклі.
   c) Видалення слів, що починаються з цифри:
   – відновлюємо S з S1;
   – готуємо рядок S2, в якому будемо зберігати ті слова, що не починаються з цифри;
   – переглядаємо рядок S по словах (виділяємо частину рядка до пробілу). Якщо слово починається з цифри, не додаємо його до S2, в іншому випадку – додаємо;
   – додаємо в массив S2 символ закінчення (щоб обірвати рядок).

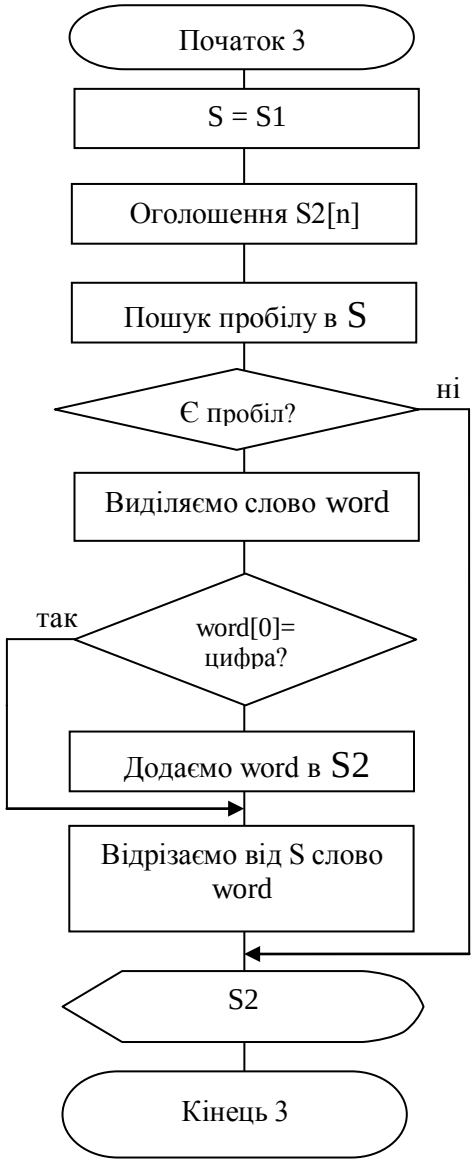
Схема роботи програми

Розробимо загальну схему роботи програми.
Для кожної з частин доцільно розробити окрему схему.


Початок
Введіть рядок:
S
1
Частина 1.
Отримання інвертор-
ваного рядка
2
Частина 2
Виділення
окремих слів
3
Частина 3
Видалення слів, що
починаються з цифри
Кінець


Початок 1
n – довжина S
Оголошення масиву
S1[n]
i=0, …, n-1
S1[i]=S[n-i-1]
S1[n]='\0'
S1
Кінець 1


Початок 2
S1 = S
Пошук пробілу в S
Є слова?
ні
Виділяємо чергове
слово word
word
Відрізаємо від S
слово word
Кінець 2


Початок 3
S = S1
Оголошення S2[n]
Пошук пробілу в S
Є пробіл?
ні
Виділяємо слово word
так
word[0]=
цифра?
Додаємо word в S2
Відрізаємо від S слово
word
S2
Кінець 3

Лістинг програми

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main()
{
  cout << "\n Input String:  ";
  char s[80];
  gets(s);
  int n=strlen(s);// довжина рядка
  char *s1 = new char[n];
  for (int i=0; i<n; i++)// перевертаємо рядок
       s1[i]=s[strlen(s)-i-1];
  s1[n]='\0';
  cout << "\n Reverse String:  " << s1;
            // виділяємо слова у реченні
  strcpy(s1,s);   // запамятовуємо початкове речення
  cout <<"\n\n\n Words in string:  \n";
  char *word;
  word = strtok(s, " " );
  while (word != NULL)
  {
       printf("\n\t%s",word);
       word=strtok(NULL, " ");
  }
  // ще раз виділяємо слова, але не виводимо ті, що починаються з цифри
  strcpy(s,s1);   // відновлюємо початкове речення
  cout <<"\n\n\n String without first digits:  \n";
  char *s2 = new char[n];
  strcpy(s2,"");
  word = strtok(s, " " );
  while (word != NULL)
  {
       if (strpbrk(word,"0123456789")==NULL)
            strcat(s2,word);
       strcat(s2," ");
       word=strtok(NULL, " ");
  }
  strcat(s2,"\0");
  printf("\n\t%s",s2);
  delete[] s1;
  delete[] s2;
  _getch();
  return 0;
}
```

Результат роботи програми

```
Input String:  HELO: aaaaaa 2bbbbbbb ccccccc 3ddddddd yy!
Reverse String:  !yy ddddddd3 ccccccc bbbbbbb2 aaaaaa :OLEH

Words in string:

        HELO:
        aaaaaa
        2bbbbbbb
        ccccccc
        3ddddddd
        yy!

String without first digits:

        HELO: aaaaaa  ccccccc  yy!
```

Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом, з функціями стандартних бібліотек для роботи з рядками і символами.
2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програму на своєму комп'ютері.
3. Розробити власну програму, які реалізує індивідуальне завдання.
4. Підготувати звіт, який включатиме:
   - варіант і текст індивідуального завдання;
   - формалізацію завдання (включаючи математичну модель задачі);
   - схему роботи програми;
   - лістинг програми;
   - роздруківку трьох контрольних прикладів (вигляд екрану з результатами – різні значення вхідних даних);
   - висновки.

Варіанти індивідуальних завдань

Задача 1. З клавіатури вводиться текстовий рядок. Розробити програму, яка реалізує вказані дії.

1	а) підраховує кількість слів, які мають непарну довжину; б) виводить на екран частоту входження кожної літери; в) видаляє текст, що розміщено в круглих дужках.
2	а) перевіряє, чи співпадає кількість відкритих і закритих дужок у введеному рядку (перевірити для круглих та квадратних дужок); б) виводить на екран найдовше слово; в) видаляє всі слова, що складаються тільки з латинських літер.
3	а) підраховує кількість різних слів, що входять до заданого тексту; б) виводить на екран кількість використаних символів; в) видаляє всі слова, що мають подвоєні літери.
4	а) підраховує кількість слів у тексті; б) виводить на екран слово, що містить найбільшу кількість голосних літер; в) видаляє з тексту всі непотрібні пробіли.
5	а) підраховує кількість розділових знаків у тексті; б) виводить всі слова, що мають парну кількість літер; в) міняє місцями першу і останню літери кожного слова.
6	а) підраховує кількість великих літер у тексті; б) виводить на екран слова, що мають найменшу кількість літер; в) видаляє всі слова, що починаються з малої літери.
7	а) підраховує кількість чисел у тексті (не цифр, а саме чисел); б) виводить на екран всі слова, що складаються тільки з латинських літер; в) видаляє кожне друге слово.

8	а) підраховує кількість цифр у тексті; б) виводить на екран слова, що починаються з приголосних літер; в) знищує всі слова, які починаються і закінчуються за одну й ту ж літеру.
9	а) підраховує кількість слів у тексті, які закінчуються на голосну літеру; б) виводить на екран всі слова, довжина яких менша п'яти символів; в) видаляє всі слова, які містять хоча б одну латинську літеру.
10	а) підраховує кількість слів у тексті, які починаються з голосної літери; б) виводить на екран всі слова, що мають непарну кількість приголосних літер; в) видаляє всі числа з тексту.
11	а) замінює всі великі літери, що входять до тексту на відповідні малі; б) виводить на екран найдовше слово; в) видаляє всі слова, що містять непарну кількість приголосних літер.
12	а) кількість слів, які містять однакову кількість голосних і приголосних літер; б) виводить на екран найдовше слово; в) видаляє з тексту всі слова-паліндроми.
13	а) виводить всі символи, які розташовані після першого символу ":"; б) підраховує кількість речень, що містять непарну кількість слів; в) видаляє з тексту всі слова, які розташовані після ком.
14	а) підраховує кількість слів у кожному реченні; б) виводить на екран найдовше речення; в) видаляє всі слова, передостання літера яких голосна.
15	а) інвертує рядок, подаючи його у зворотному вигляді; б) підраховує кількість чисел у тексті; в) видаляє всі слова, що починаються з голосних літер.

***** ***Задача 2.*** Зловмисник не викраде інформацію, про існування якої він не підозрює – ось гасло стеганографії. Карл – геть не зловмисник, однак він найближчими днями святкуватиме свій день народження, тому Аліса і Боб повинні обговорити подарунок, який вони вирішили разом презентувати Карлу.

На шляху їх благородних бажань стала одна перешкода – Боба поселили в гуртожитку в одній кімнаті з Карлом. Переймаючись тим, що Карл може ненароком побачити, про що чатяться Боб з Алісою, Боб вирішив скористатись здобутками стеганографії, приховуючи свої справжні повідомлення, керуючись:

- кожне *k*-те повторення голосної та кожне *l*-те повторення приголосної літери кирилиці становлять літери прихованого повідомлення;

- кожна зустріч латинської *літери* 1, вставленої в тексті замість аналогічної літери кирилиці, позначає, що до приблизної вартості подарунку необхідно додати десять гривень, а кожна латинська *літера* 2 – одну гривню.

Напишіть програму, яка допоможе Алісі читати приховані повідомлення Боба.

Варіант	*l*	*k*	*літера* 1	*літера* 2
1	4	6	a	y
2	3	8	x	i
3	5	7	i	e
4	2	5	y	a
5	4	4	a	x
6	3	6	x	o
7	5	8	i	y
8	2	7	y	i

Варіант	*l*	*k*	*літера* 1	*літера* 2
9	4	5	a	e
10	3	4	x	a
11	5	6	i	x
12	2	8	y	o
13	4	7	a	e
14	3	5	x	i
15	2	4	i	a

Контрольні питання

1. Який заголовочний файл використовується у мові С для роботи з символами?
2. Наведіть приклади використання функцій обробки символів.
3. Які функції існують для введення і виведення символів?
4. Наведіть основі функції введення\виведення рядків і їх призначення. Який заголововчний файл містить опис цих функцій?
5. Чим відрізняються різні функції введення\виведення рядків? Наведіть приклади їх використання.
6. Який заголовний файл використовується у мові С для обробки рядків?
7. Назвіть основні функції роботи з рядками.
8. Як можна оголосити рядок? Наведіть різні способи оголошення та ініціалізації рядків.
9. Для чого у рядках використовується завершуючий символ?
10. Як отримати довжину рядка?
11. Як компілятор мови С контролює вихід за межі масиву символів?
12. Як можна запрограмувати виділення слів у введенному реченні?
13. Яким чином можна об’єднати рядки символів та виділити підрядки за певними ознаками? Наведіть приклади.

ЛАБОРАТОРНА РОБОТА №6

ФУНКЦІЇ

Мета роботи

- Дослідити принципи побудови функцій користувача, основні складові функцій, прототипи, оголошення та опис функцій.
- Дослідити правила розробки рекурсивних функцій.
- Навчитися розробляти і використовувати на практиці власні функції для вирішення завдань, які стоять перед розробником програм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Поняття функції та її склвдові

Функція – поіменована послідовність описів і операторів, яка виконує деяку закінчену послідовність дій. Будь-яка функція складається із заголовка (оголошення функції) і тіла (визначення функції).

Оголошення функції описує її *прототип* (іноді кажуть "сигнатура").

Прототип функції оголошується наступним чином:

```
ТипПовернення Ім'яФункції (СписокОголошенихПараметрів);
```

Тут *ТипПовернення* – тип даних, що повертається функцією. Якщо він не зазначений, то за замовчуванням вважається, що повертається тип *int*.

СписокОголошенихПараметрів задає тип і ім'я кожного з параметрів функції, розділених комами (ім'я параметра можна опускати). Список параметрів функції може бути порожнім. Приклади прототипів функцій:

```
double max (double par1, double par2);
int swap(int, int) ;
void func() ;
```

Визначення функції складається з її *заголовка* і власне *тіла*, вкладеного у фігурні дужки і такого, що має смислове навантаження. Якщо функція повертає будь-яке значення, в тілі функції обов'язково повинен бути присутнім оператор повернення з параметром того ж типу.

```
ТипПоверн Ім'яФункції  (СписокОголошенихПараметрів)
{
  // тіло функції
}
```

Виклик функції – вказівка ідентифікатора функції (її імені), за яким в круглих дужках слідує список *аргументів*, розділених комами.

```
double maxValue = max (2.5, 1235.2);
int j = swap(10, 2);
func();
```

Функції, що не повертають значення

Це функції типу *void* – ті, що не повертають значення – можуть розглядатися, як деякий різновид команд, реалізований особливими програмними операторами. Оператор *func();* виконує функцію *void func()* , тобто передасть керування функції, доки не виконаються усі її оператори. Коли функція поверне керування в основну програму, тобто завершить свою роботу, програма продовжить своє виконання з того місця, де розташовується наступний оператор за оператором *func().*

Якщо функція повертає значення типу *void*, то її виклик слід організовувати так, щоб значення, яке повертається, не використовувалося. Тобто, таку функцію не використовують у правій частині виразу.

Рекурсивні функції

Рекурсія – це спосіб організації обчислювального процесу, при якому функція в ході виконання операторів звертається сама до себе.

Функція називається рекурсивною, якщо під час її роботи можливий повторний її виклик безпосередньо (прямий виклик) або шляхом виклику іншої функції, в якій міститься звернення до неї (непрямий виклик).

Прямою (безпосередньою) рекурсією називається рекурсія, при якій всередині тіла деякої функції міститься виклик тієї ж функції.

```
void fn(int i) {    ... fn(i);  ... }
```

Непрямою рекурсією називається рекурсія, що здійснює рекурсивний виклик функції шляхом ланцюга викликів інших функцій. При цьому всі функції ланцюга, що здійснюють рекурсію, вважаються рекурсивними.

```
void fnA(int i){
     ...  fnB(i); ...
}
void fnB(int i) {
     ... fnC(i); ...
}
void fnC(int i) {
     ... fnA(i); ...
}
```

Для ілюстрації рекурсії наведемо функцію обчислення *n!*:

```
double fact(int n)
{
    if (n<=1) return 1;
    return (fact(n-1)*n);
}
void main()
{   int n;
    double value;
    printf("N=");
    scanf("%d",&n);
    value=fact(n);
    printf("%d! = %.50g",n,value);
    getch();
}
```

Аргументи функції за замовчуванням

C++ допускає при виклику функцій опускати деякі її параметри. Досягається це зазначенням в прототипі функції значень аргументів за замовчуванням. Наприклад, функція, прототип якої наведено нижче, може при виклику мати різний вигляд в залежності від ситуації.

```
  // Прототип функції:
void ShowInt(int i,bool Flag=true,char symbol='\n');
  // Виклик функції ShowInt:
ShowInt(A, false, 'a');
ShowInt(B, false);
ShowInt(C);
```

Передавання параметрів у функцію

Механізм передавання параметрів є основним способом обміну інформацією між функціями. Параметри, перераховані в заголовку опису функції, називаються *формальними параметрами (або просто параметрами)*, а записані в операторі виклику функції – *фактичними параметрами* (*або аргументами*).

Існує два способи передачі параметрів у функцію: за значенням і за адресою.

При *передаванні за значенням* в стек заносяться копії значень аргументів, і оператори функції працюють з цими копіями. Доступу до початкових значень параметрів у функції немає, а, отже, немає і можливості їх змінити.

При *передаванні за адресою* в стек заносяться копії адрес аргументів, а функція здійснює доступ до комірок пам'яті за цими адресами і може змінити значення аргументів:

```
#include <iostream.h>
void f(int i, int* j, int& k);
int main()
 {
      int i=1,j=2,k=3;
      cout <<"i j k\n";
      cout<<i<<" "<<j<<" "<<k<<'\n'; // Результат: 1 2 3
      f(i,&j,k);
      cout<<i<<" "<<j<<" "<<k;        // Результат: 1 3 4
      return 0;
 }
 void f(int i, int* j, int& k)
 {
      i++; (*j)++; k++;
 }
```

Перший параметр (i) передається за значенням. Його зміна в функції не впливає на вихідне значення. Другий параметр (j) передається за адресою за допомогою покажчика, при цьому для передаванні у функцію адреси фактичного параметра використовується операція взяття адреси, а для отримання його значення в функції потрібна операція розіменування. Третій параметр (k) передається адресою за допомогою посилання.

При передаванні за посиланням у функцію передається адреса зазначеного при виклику параметра, а всередині функції всі звернення до параметру неявно розіменовуються. Використання посилань замість покажчиків, по-перше, покращує читабельність програми, позбавляючи від необхідності застосовувати операції одержання адреси та розіменування; і, по-друге, не вимагає копіювання параметрів, що важливо при передаванні структур даних великого обсягу.

Якщо потрібно заборонити зміну параметра усередині функції, використовуєтьсямодифікатор *const*:

```
int f(const char*);
char* t(char* a, const int* b);
```

Масиви як параметри функції

Усі параметри, за винятком параметрів типу покажчик та масив, передаються за значенням. Це означає, що під час виклику функції їй передаються тільки значення змінних. Сама функція не в змозі змінити цих значень у функції, що їх викликає.

Одновимірний масив як параметр

Ім’я масиву є покажчиком на його нульовий елемент, тому у функцію масиви передаються за покажчиком. Кількість елементів масиву може передаватися окремим параметром.

Приклад 1. Програма підраховує, в якому з двох масивів кількість додатних елементів більша.

```
int n_posit(int* mas, int n);
int main()
{ int n,i;
  cout<<"Input count of symbols:  ";
  cin>>n;
  int *A = new int[n];
  int *B = new int[n];
  for (i=0; i<n; i++)// генеруємо елементи масивів
  {     A[i] = rand()%100 - 50;
        B[i] = rand()%200 - 100;
  }
  cout << "\n\nA:  ";// виводимо масиви на екран
  for (i=0; i<n; i++) cout << "   "<<A[i];
  cout << "\n\nB:  ";
  for (i=0; i<n; i++) cout << "   "<<B[i];
        // порівнюємо кількість додатних елементівтів
  if (n_posit(A,n) > n_posit(B,n))
        cout << "\n\nFirst array A is winner!";
  else  if (n_posit(A,n) < n_posit(B,n))
              cout << "\n\nSecond array B is winner!";
        else  cout << "\n\nEqual count!";
  _getch();
  return 0;
}
int n_posit(int* mas, int n)
{ int count = 0;
```

```
    for (int i=0; i<n; i++)
          if (mas[i]>0) count++;
    return count;
}
```

Результат роботи програми:

```
Input count of symbols:  10

A:     -9   -16   19   28   12   -45   31   11   45   -23
B:     -33   0   24   58   -36   45   -73   -9   42   -64
First array A is winner!
```

Рядок (масив символів) як параметр функції

На відміну від передавання у функцію звичайного масиву, для рядка немає необхідності окремим параметром передавати розмірність масиву. Можна визначати кінець рядка за нуль-символом ('\0').

Приклад 2. Програма отримує рядок і виводить його посимвольно, вставляючи між сиволами знаки підкреслювання.

```
void outMyText(char* str);
int main()
{ cout<<"\n  InString:  ";;
  char str[80];
  gets(str);
  outMyText(str);
  _getch();
  return 0;
}
void outMyText(char* s)
{ cout << "\n\n\n  OutString:  ";
  int i = 0;
  while(s[i] != '\0')
  {     cout << "__" << s[i];
        i++;
  }
}
```

Результат роботи програми:

```
InString:  Hello, students!


OutString:  __H__e__l__l__o__,__ __s__t__u__d__e__n__t__s__!
```

Двовимірний масив в якості параметра функції

Приклад 3. Виведення масиву на екран по рядках.

```
void printArray(int** mas, int n, int m);
int main()
{ int n,m,i;
  cout<<"\n  Input n m:  ";      cin>>n>>m;
  int** A = new int*[n];
              // генеруємо елементи масивів
  for (i=0; i<n; i++)
  {     A[i] = new int[m];
        for (int j=0; j<m; j++)  A[i][j] = rand()%200 - 100;
```

```
    }
    printArray(A,n,m);
    _getch();
    return 0;
  }
  void printArray(int** mas, int n, int m)
  { cout << "\n\nARRAY:\n\n";
    for (int i=0; i<n; i++)
    {     cout<<"\n";
          for (int j=0;j<m;j++)    printf("%6i",mas[i][j]);
    }
  }
```

Результат роботи програми:

```
 Input n m:  5  10

ARRAY:

   -59   -33    34     0    69    24   -22    58    62   -36
     5    45   -19   -73    61    -9    95    42   -73   -64
    91   -96     2    53    -8    82   -79    16    18    -5
   -53    26    71    38   -31    12   -33    -1   -65    -6
     3   -89    22    33   -27   -36    41    11   -47   -32
```

Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом.
2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програми на своєму комп'ютері.
3. Розробити власні програми, які реалізу.ть індивідуальні завдання.
4. Результати розробки представити у звіті, який матиме такі складові:
   – варіант, завдання;
   – загальна схема роботи програми (для першого завдання – без детальних схем функцій, для другого – включаючи схеми функцій);
   – схема взаємозв'язків функцій у програмі;
   – лістинг програми (з коментарями);
   – результати виконання програми;
   – висновки.

Варіанти індивідуальних завдань

***Задача 1*.** Розробити програму, яка працюватиме наступним чином. Головна функція програми повинна викликати інші функції, створені за результатами попередніх лабораторних робіт (сформувати повне завдання):

– функція, яка повертає випадкове число з вказаного діапазону (границі інтервалу – аргументи функції) – лабораторна робота №2 (задача 1);
– функція, яка виконує завдання лабораторної роботи №3 (задача 1);
– функція, яка виконує завдання лабораторної роботи №4 (задача 1);
– функція, яка реалізує меню для керування роботою програми;
– інші функції, необхідні для виконання.

Задача 2. Розробити програму для реалізації індивідуального завдання.

<table>
<tr><td>1</td><td>Написати функцію, яка виводить на екран суцільний прямокутник, створений із заданих символів C (наприклад, '#'), сторони прямокутника визначаються змінними W та H. Наприклад, якщо C='*', W=5, H=5, то функція виведе показане зображення.
*****
*****
*****
*****
*****</td></tr>
<tr><td>3</td><td>Напишіть функцію, яка виводить ціле число між 1та 32767 і друкує це число як послідовність цифр, між якими стоять два пробіли. Наприклад, ціле число 4562 повинно бути надруковане так: 4 5 6 2.</td></tr>
<tr><td>2</td><td>Функція отримує час у вигляді трьох цілих аргументів (години, хвилини, секунди) і повертає кількість секунд з моменту, коли на годиннику було 0 годин. Використайте цю функцію для підрахунку часу в секундах між двома моментами часу.</td></tr>
<tr><td>3</td><td>Напишіть функцію, яка виводить ціле число між 1та 32767 і друкує це число як послідовність цифр, між якими стоять два пробіли. Наприклад, ціле число 4562 повинно бути надруковане так: 4 5 6 2.</td></tr>
<tr><td>4</td><td>Ціле число називається досконалим (рос. совершенным), якщо сума його дільників, включаючи 1 (але не саме число) , дорівнює цьому числу. Наприклад, число 6 = 1+2+3 є досконалим. Напишіть програму perfect, яка визначає, чи є число досконалим. Використайте цю функцію у програмі, яка знаходить і друкує усі досконалі числа з діапазону від 1 до 1000.</td></tr>
<tr><td>5</td><td>Ціле число називається простим, якщо воно ділиться на 1 і на самого себе. Наприклад, числа 2,3,5 і 7 є простими, а 2,6,8 і 9 – ні. Напишіть функцію, яка визначає, чи є число простим. Використайте цю функцію у програмі, яка знаходить і друкує усі прості числа у діапазоні від 1 до 10000.</td></tr>
<tr><td>6</td><td>Напишіть функцію, що отримує ціле значення і повертає число з оберненим порядком цифр. Наприклад, для 7631 функція повинна повернути 1367.</td></tr>
<tr><td>7</td><td>Напишіть функцію, яка вводить бали, зароблені студентом, а повертає оцінку за шкалою ECTS:
<table>
<tr><td>Сума балів</td><td>90 - 100</td><td>82-89</td><td>74-81</td><td>64-73</td><td>60-63</td><td>35-59</td><td>0-34</td></tr>
<tr><td>Оцінка ECTS</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>FX</td><td>F</td></tr>
</table></td></tr>
<tr><td>8</td><td>Напишить функцію distance, яка обраховує відстань між двома точками з координатами (x1, y1) і (x2, y2). Усі числа і повернене значення повинні бути дійсними. Використайте цю функцію у програмі, яка обчислює площу трикутника за формулою Герона.</td></tr>
<tr><td>9</td><td>Напишіть програму, яка допоможе школяру вивчити таблицю множення. Згенеруйте два додатних однорозрядних числа (окрема функція). Програма виводить, наприклад, питання: "Скільки буде 5 на 6?".
Школяр повинен відповісти. Якщо відповідь правильна, програма друкує
"Молодець! Дуже добре!"
і після цього задаю наступне питання на множення. Якщо відповідь неправильна, програма друкує: "Невірно! Спробуйте знову ...", до тих пір, поки відповідь не буде правильною. Розроблювана функція повинна отримувати три числа: два множники та число-віповідь школяра, і друкувати потрібну фразу.</td></tr>
<tr><td>10</td><td>Напишіть функцію muliply для двох цілих, яка визначатиме, чи кратне друге число першому, Функція повинна отримувати два цілих аргументи і повертати 1 (true), якщо друге число кратне першому, і 0 (false) в іншому випадку. Використайте цю функцію у програмі, яка вводить серію пар цілих чисел.</td></tr>
</table>

11	Напишіть програму, яка грає у гру "Вгадай число" наступним чином: ваша програма "задумує" число (випадкове число у діапазоні від 1 до 1000), яке треба вгадати. Далі програма друкує: *У мене є число між 1 та 1000. Відгадайте і введіть ваше число…* Далі гравець вводить перше число. Програма відповідає однією з фраз: *Чудово! Ви вгадали число! Будете грати далі?* *Занадто мале. Спробуйте ще раз.* *Занадто велике. Спробуйте ще раз.* При реалізації гри необхідно написати функцію, яка приймає два числа: "задумане" і відповідь гравця, а після аналізу друкувати одну з фраз.
12	Написати функцію, яка виводить на екран трикутник, створений із заданих символів C (наприклад, '#'), основа трикутника визначається змінною W. Наприклад, якщо C='*', W=5, то функція виведе показане зображення. ***** **** *** ** *
13	Напишіть програму, яка буде використовувати функцію, що обчислює скалярний добуток двох векторів дійсних чисел, що мають однакову розмірність.
14	Розробити функцію, яка змінює значення двох заданих дійсних змінних. Перша змінна повинна отримати значення суми початкових даних, а друга – значення їх різниці.
15	Напишіть програму, яка вводить довільні натуральні числа і визначає, чи є введене число паліндромом. Для визначення, чи є число паліндромом, написати окрему функцію, яка приймає число і повертає true або false. Число називається поліндромом, якщо воно сліва направо і справа наліво читається однаково. Наприклад, числа 232, 7667 і 34543 є поліндромами, а числа 123 і 45 – ні.

*** *Задача 3.*** В програмі святкування дня народження кафедри захисту інформації заплановано 7 номерів. Аліса, Боб і Карл приймають участь у цих виступах. Оскільки вони – люди творчі, тому виступи, які вони бачать до свого впливають на рівень їх натхнення. Для ідеального виступу Алісі необхідно 8 одиниць натхнення, Бобу – 9, Карлу – 7. В інших випадках якість їх виступу буде пропорційно менша за можливий максимум.

Оскільки Аліса, Боб та Карл мають різні вподобання, тому однакові номери надихають їх по-різному.

Студент	Музика	Лірика	Гумор
Аліса	4	2	3
Боб	3	3	2
Карл	1	2	4

Внаслідок отримання чудового подарунку від Аліси та Боба на власний день народження Карл ще до початку вистави має 3 одиниці натхнення.

Для вистави Аліса приготувала гру на кобзі, Боб написав ліричного вірша про кафедру і студентське життя, а Карл бере участь у жартівливій сценці про викладачів.

Написати програму, яка визначатиме якість виступів (у відсотках) та рівень натхнення Аліси, Боба та Карла після святкування дня народження

кафедри, за умови, що їх власні виступи їх не надихають (оскільки вони себе не бачать, коли виступають).

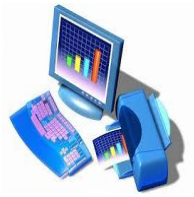
Варіант	Номери						
	1	2	3	4	5	6	7
1	Музика	Гумор	*Карл*	*Аліса*	Лірика	*Боб*	Гумор
2	Лірика	Музика	Гумор	Лірика	*Аліса*	*Карл*	*Боб*
3	Гумор	Лірика	Музика	*Боб*	Гумор	*Аліса*	*Карл*
4	Лірика	Гумор	Лірика	Музика	*Боб*	*Аліса*	*Карл*
5	Музика	Лірика	*Аліса*	Гумор	Лірика	*Боб*	*Карл*
6	Музика	Лірика	*Боб*	*Аліса*	Лірика	*Карл*	Гумор
7	Лірика	Гумор	*Карл*	Музика	*Боб*	Гумор	*Аліса*
8	Гумор	Музика	Гумор	*Боб*	*Аліса*	Лірика	*Карл*
9	Лірика	Лірика	Музика	*Карл*	*Боб*	*Аліса*	Гумор
10	Музика	Гумор	Лірика	Гумор	*Боб*	*Карл*	*Аліса*
11	Музика	Музика	*Карл*	*Аліса*	Лірика	*Боб*	Гумор
12	Лірика	Музика	*Боб*	Гумор	*Аліса*	Лірика	*Карл*
13	Гумор	Гумор	*Боб*	*Карл*	Лірика	*Аліса*	Музика
14	Лірика	Музика	Гумор	Музика	*Карл*	*Аліса*	*Боб*
15	Музика	Лірика	Музика	Гумор	*Боб*	*Карл*	*Аліса*

Контрольні питання

1. Що називають функцією? Що таке опис функції?
2. Що таке прототип функції? Де його розміщують у програмі?
3. Які символи оточують тіло функції? Які символи оточують аргументи функції?
4. Чи обов’язково в прототипах функцій вказувати ідентифікатори змінних?
5. Як відбувається звернення до функції?
6. Яке ключове слово використовується в заголовку функції, щоб показати, що функція не повертає нічого і немає жодного параметра?
7. Чи кожна функція повинна мати оператор повернення?
8. Нведіть специфікатори класу пам’яті.
9. Що таке локальні змінні? Що означає специфікатор пам’яті *auto*?
10. Що таке глобальні змінні? Що означає специфікатор пам’яті *external*?

11. Що означає специфікатор пам’яті *static*? Чи можуть статичні змінні бути локальними?
12. Чи можна у функцію передавати параметри за замовчуванням?
13. Що таке фактичні параметри функції?
14. Що таке формальні параметри?
15. Чим відрізняються фактичні параметри від формальних?
16. Чи можуть ідентифікатори фактичних і формальних параметрів співпадати?
17. Чи обов’язково кількість фактичних і формальних параметрів повинні співпадати?
18. Що таке рекурсивна функція? Навести приклад.
19. Що означає, що параметри передаються у функцію за значенням?
20. Що означає, що параметри передаються у функцію за покажчиком?
21. Як передати у функцію масив? Навести приклад.
22. Як передати у функцію рядок символів? Навести приклад.
23. Що означають параметри головної функції програми?
24. Чи може глобальна змінна бути розташована у тілі програми?
25. Чи можна у середині однієї функції оголошувати іншу функцію?
26. Що таке вбудовані функції? Коли вони використовуються?

ЛАБОРАТОРНА РОБОТА №7



СТРУКТУРИ. МАСИВИ СТРУКТУР

Мета роботи

- Оволодіти практичними навичками зі створення та використання структур, передавання структур у функції в якості параметрів.
- Навчитись використовувати масиви структур та здійснювати операції з полями структур.
- Навчитися складати програми з використанням структур.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Оголошення структури

Структури дозволяють об'єднувати в єдиному об'єкті сукупність значень, які можуть мати різні типи. Оголошення структури здійснюється за допомогою ключового слова *struct*. Синтаксис опису структури :

```
struct [ім'я_структури] {
    тип1 елемент1;
    .............
    типN елементN;
}  [список описів];
```

Приклад структури – представлення поняття "дата", що складається з декількох частин: число (день, місяць, рік), назва тижня та місяця:

```
struct date {
     int day ;
     int month ;
     int year;
     char day_name[15];
     char mon_name[14];
} arr[100],*pd, data, new_data;
```

Тут *data*, *new_data* – змінні типу *date*; *pd* – покажчик на тип *date; arr* – масив зі 100 елементів, кожний з яких має тип *date*.

Доступ до полів структури

Доступ до полів структури забезпечується операторами вибору: **.** (прямий селектор) та **->** (непрямий селектор), наприклад,

```
struct mystruct {
       int i;
       char str[21];
       double d;
} s,*sptr=&s;
s.i =3;
sptr->d = 1.23;
```

Зауваження

Для змінних одного і того ж самого структурного типу визначена операція присвоювання, при цьому здійснюється поелементне копіювання значень полів.

```
data=new_data;
```

Для порівняння структур необхідно перевіряти рівність відповідних полів цих структур.

```
struct point {
       float x,y;
       char c;
} point1,point2;
if (point1.x==point2.x&&point1.y==point2.y&&point1.c==point2.c){
       /* … */
};
```

Масиви структур

Масивами структур, елементи якого мають структурований тип, можна оперувати, як звичайними масивами простих типів. Приклад:

```
typedef struct Date
{
        int d; /* день */
        int m; /* місяць */
        int y; /* рік */
} Date;
Date arr[100];
```

Доступ до полів структури аналогічний доступу до звичайних змінних, при цьому у квадратних дужках вказують індекс номеру елемента:

```
arr[25].d=24; arr[12].m=12;
```

Об'єднання (union)

Об'єднання дозволяють в різні моменти часу зберігати в одному об'єкті значення різного типу. В кожний момент часу об'єднання може зберігати значення тільки одного типу з набору. Контроль за тим, значення якого типу зберігається в даний момент, покладається на програміста.

```
union sign
{   int svar;
    unsigned uvar;
} number;
```

```
union
{   char *a,b;
    float f[20];
} var;
```

Бітові поля

Бітові поля (*bit fields*) – особливий вид полів структури. Вони дають можливість задавати кількість бітів, в яких зберігаються елементи цілих типів. При оголошенні бітового поля вслід за типом елемента ставиться двокрапка (**:**) і вказується цілочисельна константа, яка задає розмір поля (кількість бітів). Розмір поля – константа в діапазоні між 0 і числом бітів, яке використовується для зберігання даного типу даних.

```
struct bit_field {
       int bit_1   : 1;
       int bits2_5 : 4;
       int bit_6   : 1;
       int bits7_16: 10;
} bit_var;
```

Приклади програм з використанням структур

Приклад 1. Сформувати список про зареєстровані перездачі заборгованостей студентів: ПІБ, предмет, кількість перездач. Роздрукувати список за заданою ознакою і видати його на екран дисплея.

Лістинг програми

```
#include <stdio.h>
#include <conio.h>
#define n 3
void main()
{    int i,f=0,obrz;
     struct
     {    char FIO[20];
          char predmet[20];
          int kol_peresdach;
     } dolg[n];
     for(i=0;i<n;i++)
      {    printf("\nBedite FIO           ");
           scanf("%s",&dolg[i].FIO);
           printf("Bedite predmet       ");
           scanf("%s",&dolg[i].predmet);
           printf("Bedite kol_peresdach ");
           scanf("%d",&dolg[i].kol_peresdach);
      }
     printf("\nBedite obrz      ");
     scanf("%d",&obrz);
     for(i=0;i<n;i++)
     {  if(dolg[i].kol_peresdach==obrz)
        {  f++;
           printf("\n%15s\t%15s\t%3i ",
                dolg[i].FIO,dolg[i].predmet,dolg[i].kol_peresdach);
        }
     }
     if(f==0)printf("\n Error");
     getch();
  }
```

Результати роботи програми:

```
Bedite FIO            Petrov
Bedite predmet        Programming
Bedite kol_peresdach 3

Bedite FIO            Sidorov
Bedite predmet        Mathematic
Bedite kol_peresdach 3

Bedite FIO            Kaplun
Bedite predmet        Informatic
Bedite kol_peresdach 2

Bedite obrz      3

          Petrov      Programming   3
         Sidorov       Mathematic   3
```

Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом, з функціями стандартних бібліотек для роботи з рядками і символами.
2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програми на своєму комп'ютері.
3. Розробити власні програми, які реалізують індивідуальне завдання.
4. Підготувати загальний звіт для всіх задач , який включатиме:

– варіант і текст індивідуального завдання;
– загальну схему роботи програми (лише для задачі 1);
– лістинг програми;
– роздруківку результатів роботи програми у різних режимах;
– висновки.

Варіанти індивідуальних завдань

Задача 1.

1.	Описати структуру з іменем STUD, яка містить поля: NAME – прізвище та ініціали; GROUP – група; SES – оцінки з п'яти предметів (масив з п'яти елементів). Написати програму, що реалізує наступні дії окремими функціями: – введення з клавіатури даних в масив ST, що складається з N змінних типу STUD; – виведення на екран прізвищ і номерів груп для всіх студентів, середній бал яких більший за 4.0; якщо таких немає, то вивести повідомлення.
2.	Описати структуру з іменем ABIT, яка містить поля: NAME – прізвище, ініціали; GENDER – стать; SPEC – назва спеціальності; EXAM – результати вступних іспити з трьох предметів (масив з трьох елементів). Написати програму, що окремими функціями реалізує наступні дії: – введення з клавіатури даних в масив AB, що складається з N змінних типу ABIT; – виведення на екран прізвищ та назв спеціальностей для всіх абітурієнтів, що мають бал нижче, ніж прохідний, який визначається користувачем програми; якщо таких студентів немає, то вивести повідомлення про це.
3.	Описати структуру з іменем NOTE, яка містить поля: NAME – прізвище, ім'я; TEL – номер телефону; BDAY – день народження (масив із трьох чисел). Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив BLOCKNOTE, що складається з N змінних типу NOTE; – виведення на екран інформації про людей, чиї дні народження припадають на місяць, значення якого введено з клавіатури; якщо таких людей немає, то вивести відповідне повідомлення.
4.	Описати структуру з іменем AEROFLOT, яка містить поля: CITY – назва населеного пункту призначення; NUM – номер рейса; TYPE – тип літака. Написати програму, що окремими функціями реалізує наступні дії: – введення з клавіатури даних в масив AIR, що складається з N змінних типу AEROFLOT; – виведення на екран номерів рейсів і типів літаків, що вилетіли в пункт призначення, назва якого співпала з назвою, введеною з клавіатури; якщо таких рейсів немає, то вивести відповідне повідомлення.

5.	Описати структуру з іменем SKLAD, яка містить поля: NAME – назва товару; TYPE – одиниця виміру; COST – ціна одиниці товару; QUANTITY – кількість. Написати програму, що окремими функціями виконує дії: – введення з клавіатури даних в масив SHOP, що складається з N змінних типу SKLAD; – виведення на екран інформації про товар, його кількість, ціну одиниці та обчислену загальну суму на складі, назва якого вводиться з клавіатури; якщо такого немає, то вивести відповідне повідомлення.
6.	Описати структуру з іменем WORK, яка містить поля: NAME – прізвище та ініціали працівника; POS – назва посади; YEAR, MONTH – рік і місяць прийняття на роботу. Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив TABL, що складається з N змінних типу WORK; – виведення прізвища працівників, стаж роботи яких перевищує значення, введене з клавіатури; якщо таких немає, то вивести повідомлення.
7.	Описати структуру з іменем TRAIN, яка містить поля: NAZV – назва пункту призначення; NUMR – номер потягу; DATE, TIME – дата і час відправлення. Написати програму, що окремими функціями виконує дії: – введення з клавіатури даних в масив RASP, що складається з N змінних типу TRAIN; – виведення на екран інформації про поїзди, що відправляються після введеного з клавіатури дня та часу; якщо таких поїздів немає, то вивести відповідне повідомлення.
8.	Описати структуру з іменем TABLE, яка містить поля: NAZV – назва пункту призначення; NUMR – номер поїзда; DATE, TIME – дата і час відправлення. Написати програму, що окремими функціями виконує дії: – введення з клавіатури даних в масив TRAIN, що складається з N змінних типу TABLE; – виведення на екран інформації про поїзди, що направляються в пункт призначення, назва якого введена з клавіатури; якщо таких немає, то вивести повідомлення про це.
9.	Описати структуру з іменем ZNAK, яка містить поля: NAME – прізвище, ім'я; ZODIAC – знак Зодіаку; BDAY – день народження (масив із трьох чисел). Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив BOOK, що містить N змінних типу ZNAK; – виведення на екран інформації про людину, чиє прізвище введене з клавіатури; якщо таких людей немає, то вивести відповідне повідомлення.
10.	Описати структуру з іменем ITNR, яка містить наступні поля: FIRST – назва початкового пункту маршруту; FINAL – назва кінцевого пункту маршруту; NUM – номер маршруту; DISTANCE – відстань у кілометрах. Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив RT, що складається з N змінних типу ITNR; – виведення на екран інформації про маршрут, номер якого введений з клавіатури; якщо таких маршрутів немає, то вивести відповідне повідомлення.
11.	Описати структуру з іменем ITIN, яка містить поля: BEG, END – назви початкового і кінцевого пунктів маршруту; NUM – номер маршруту; DIST – відстань у кілометрах. Написати програму, що окремими функціями виконує дії: – введення з клавіатури даних в масив RT, що складається з N змінних типу ITIN; – виведення на екран інформацію про маршрути, які починаються або закінчуються в пункті, назва якого введена з клавіатури; якщо таких немає, то вивести повідомлення.

12.	Описати структуру з іменем SCHL, яка містить поля: NAME – прізвище та ім'я учня; GROUP – номер групи; SUBJ – успішність з п'яти предметів (масив з п'яти елементів). Написати програму, що окремими функціями виконує дії: – введення з клавіатури даних в масив LEARN, що складається з N змінних типу SCHL; – виведення на екран прізвищ і номерів груп для всіх студентів, що мають хоча б одну оцінку "2"; якщо таких немає, то вивести відповідне повідомлення.
13.	Описати структуру з іменем TBL, яка містить поля: NAZV – назва пункту призначення; NUMR – номер поїзда; DATE – дата відправлення; TIME – час відправлення. Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив TRAIN, що складається з N структур типу TBL; – виведення на екран інформацію про поїзди, дата відправлення яких введена з клавіатури; якщо таких поїздів немає, то вивести відповідне повідомлення.
14.	Описати структуру з іменем ABIT, яка містить поля: NAME – прізвище, ініціали; GENDER – стать; SPEC – назва спеціальності; EXAM – результати вступних іспитів з трьох предметів (масив з трьох елементів). Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив ABIT, що складається з N змінних типу ABIT; – виведення на екран прізвищ та назв спеціальностей для всіх абітурієнтів, що набрали прохідний бал, який визначається користувачем програми; якщо таких студентів немає, то вивести відповідне повідомлення.
15.	Описати структуру з іменем TOVAR, яка містить наступні поля: NAME – назва товару; TYPE – одиниця виміру товару; SORT – сорт товару; QUANTITY – кількість одиниць товару; COST – ціна. Написати програму, що окремими функціями виконує наступні дії: – введення з клавіатури даних в масив SHOP, що складається з N змінних типу TOVAR; – виведення на екран інформації про товар, його кількість, ціну одиниці та обчислену загальну суму товару на складі; назва товару вводиться з клавіатури, якщо його немає, то вивести відповідне повідомлення.

Задача 2. Розробити програму, використовуючи функції, розмішені всередині структури, що описує відповідний об’єкт.

1.	Інформація по N заводах міста задається рядком такого вигляду: прізвище, середній вік, спеціальність, середній оклад. Ввести інформацію по заводах, порахувати кількість слюсарів та токарів. Надрукувати значення та номери заводів, де середній вік вище 35 років.
2.	Багаж пасажира характеризується кількістю речей і загальною вагою. Ввести інформацію про N пасажирів і визначити, чи є серед пасажирів такий, у якого найбільший багаж по кількості і за вагою.
3.	Інформація по N заводах міста задається рядком такого вигляду: прізвище, середній вік, спеціальність, середній оклад. Ввести інформацію по заводах, порахувати середній оклад по всіха заводах. Порахувати кількість заводів, де середній оклад по заводу вище середнього по всіх заводам. Надрукувати це значення і вивести інформацію по цих заводах.

4.	Є інформація за підсумками іспитів в інституті, всього в списку N чоловік. По кожному з студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити і надрукувати кількість і прізвища відмінників та кількість і прізвища студентів, які мають хоча б одну двійку.
5.	Є інформація за підсумками іспитів в інституті, всього в списку N чоловік. По кожному із студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити, обчислити і надрукувати середній бал з кожної дисципліни. Надрукувати прізвища студентів, у яких бал по кожному з предметів вище середнього по цьому предмету.
6.	У населеному пункті проживає N чоловік. Про кожного відомі прізвище, вік, стать. Ввести інформацію про жителів даного пункту і порахувати кількість жінок і чоловіків, вивести інформацію про тих, кого більше.
7.	У населеному пункті проживає N чоловік. Про кожного відомі прізвище, вік, стать. Ввести інформацію про жителів даного пункту і порахувати кількість жінок і чоловіків, вивести інформацію про середній вік чоловіків, надрукувати прізвища тих чоловіків, чий вік нижче середнього.
8.	Картотека відеотеки організована у вигляді масиву структур з полями: назва фільму, вартість, режисер. Ввести інформацію по відеотеці і вивести інформацію про фільми одного режисера. Вивести інформацію про всі фільми за зростанням вартості.
9.	Картотека відеотеки організована у вигляді масиву структур з полями: назва фільму, вартість, режисер. Ввести інформацію по відеотеці і вивести інформацію про фільми, вартість яких більша, ніж середня, максимальна і мінімальна вартість.
10.	Є інформація про N членів спортивної секції: прізвище, вік, зріст. Ввести інформацію про кожного. Надрукувати прізвище найвищого. Надрукувати інформацію про тих, чий вік нижче середнього, а зріст вище середнього по секції.
11.	Є інформація про N членів спортивної секції: прізвище, вік, зріст. Ввести інформацію про кожного. Вивести інформацію про всіх спортсменів у таблиці. Надрукувати прізвища та вік тих, чий вік вище середнього.
12.	Є інформація за підсумками іспитів в інституті всього в списку N чоловік. По кожному із студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити і надрукувати кількість і прізвища студентів, які отримали на іспиті дві п'ятірки і одну четвірку.
13.	Є інформація про N учасників спортивних змагань з п'ятиборства. Про кожного учасника відома наступна інформація: прізвище, місце, зайняте по кожному з видів. Ввести інформацію про учасників змагань і вивести інформацію про переможця у п'ятиборстві.
14.	Є інформація про N учасників спортивних змагань з п'ятиборства. Про кожного учасника відомо: прізвище, місце, зайняте по кожному з видів. Ввести інформацію про учасників змагань і вивести інформацію про переможця у кожному виді спорту.
15.	Є інформація за підсумками іспитів в інституті, всього в списку N чоловік. По кожному із студентів є такі відомості: прізвище, оцінка з математики, оцінка з інформатики та оцінка з фізики. Ввести інформацію про іспити і надрукувати кількість і прізвища студентів, які склали іспити тільки з однією четвіркою. Надрукувати кількість і прізвища студентів, які здали інформатику з оцінкою відмінно.

.

* ***Задача 3.***

1, 5, 10, 15	Завершення першого семестру навчання поставило перед Алісою складну задачу, а саме вирішення того, що робити зі старими конспектами. Гуляючи набережною Південного Бугу вона вирішила їх збирати, а для зручності каталогізувати, позначаючи назву дисципліни, напрям (технічна, маремати чна або гуманітарна), відомості про лектора та кількість сторінок. Остерігаючись вірусів, Аліса вирішила створити кон-

	трольну суму каталогу, побайтно додаючи вміст всього каталогу, а також позначати дату, коли створена ця контрольна сума. Напишіть програму, яка допоможе Алісі у її прагненні до систематизації набутих знань.
2, 6, 11	Аліса любить шоколадки і знає, що це корисно для мозку, однак водночас шкідливо для зубів та фігури. Намагаючись слідкувати за кількістю спожитого нею шоколаду, Аліса записує назву шоколаду, частку какао в ньому, спожиту кількість за день. Напишіть програму, яка обробляючи дані Аліси, зібрані за місяць, зможе визначити середню кількість шоколаду на добу, яку споживає Аліса та її улюблений вид шоколадок, щоб Боб з Карлом знали, як їй краще віддячити у випадку, коли вони будуть користуватись її конспектом.
3, 7, 12	Боб у своєму марному прагненні дізнатись, що він більше полюбляє – поезію чи музику, збирав дані про вірші, які він читав, записуючи час читання, назву збірки, автора, а також збирав дані про музику, яку він слухав, а саме час, який він витрачав на прослуховування, жанр музики, виконавця, композитора, автора слів у випадку пісень. Після трьох місяців записів, він збагнув, що його бажання читати вірші або слухати музику обумовлюється більше його настроєм, ніж вподобаннями. А ще він усвідомив, що пісні поєднують як музику, так і поезію. Однак Боб завжди був оптимістом, тому вирішив не засмучуватись з приводу виконаної роботи щодо каталогізації, а використати її результати для того, щоб дізнатися який жанр музики він найчастіше слухає та вірші якого з авторів найчастіше манять його. Крім того, він вирішив дізнатися чи є автори слів пісень серед його улюблених поетів, а також скільки в середньому часу він витрачав на дозвілля протягом цих трьох місяців. Напишіть програму, яка позбавить Боба від "ручного" обчислення та пошуку.
4, 9, 14	Одного дня Карл усвідомив, що його жарти подобаються не всім. Вирішивши, що він найбільше дружить з Алісою та Бобом, Карл почав записувати, які жарти вони полюбляють, які не викликають в них посмішок, а які викликають осуд. Карл згрупував всі жарти на такі категорії: програмістські, студентські, загальнолюдські. Під час спостережень Карл занотовував тип жарту, реакцію Аліси, реакцію Боба та опис самого жарту. Напишіть програму, що на основі п’ятдесятиденних спостережень Карла допоможе йому визначити улюблені жанри жартів для своїх друзів.

Контрольні питання

1. Що таке структури? Який синтаксис опису структур?
2. Чим відрізняється оголошенн структур в мовах С і С++?
3. Як можна оголосити статичний та динамічний масив структур?
4. Наведіть приклад оголошення масиву змінних типу структура.
5. Як можна звернутись до полів структури? Наведіть приклади .
6. Як здійснити присвоєння структур та порівняння структур? В чому особливість цих операцій над структурами?
7. Що являють собою об’єднання? Наведіть приклади об’єднань.
8. Що таке бітові множини? Для чого їх використовують?
9. Наведіть приклади використання бітових множин.
10. Наведіть фрагмент коду для отримання двійкового представлення деякої змінної?

ЛАБОРАТОРНА РОБОТА №8

РОБОТА З ФАЙЛАМИ

Мета роботи

- Вивчити програмні засоби для роботи з файлами та потоками..
- Дослідити основні функції роботи з файлами та реалізовувати найпростіші операції з ними.
- Навчитись застосовувати у своїх програмах вхідні і вихідні текстові і бінарні файли і файлові потоки.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Поняття файлів і потоків

В комп'ютерній системі між програмою і пристроєм знаходиться щось більш загальне, ніж сам пристрій. Такий узагальнений пристрій введення або виведення (пристрій більш високого рівня абстракції) називається ***потоком***, в той час як конкретний пристрій називається ***файлом*** (файл – теж поняття абстрактне). Потоки бувають двох видів: текстові і двійкові.

Текстовий потік – це послідовність символів. У стандарті С вражається, що текстовий потік організований у вигляді рядків, кожен з яких закінчується символом нового рядка. Проте, в кінці останнього рядка цей символ не є обов'язковим. В текстовому потоці на вимогу базового середовища можуть відбуватися певні перетворення символів. Тому може і не бути однозначної відповідності між символами, які пишуться (читаються), і тими, які зберігаються в зовнішньому пристрої. Крім того, кількість тих символів, які пишуться (читаються), і тих, які зберігаються в зовнішньому пристрої, може також не співпадати із-за можливих перетворень.

Двійковий потік – це послідовність байтів, яка взаємно однозначно відповідає байтам на зовнішньому пристрої, причому ніякого перетворення символів не відбувається. Крім того, кількість тих байтів, які пишуться (читаються), і тих, які зберігаються на зовнішньому пристрої, однакова. Однак наприкінці двійкового потоку може додаватися визначена додатком кількість нульових байтів (наприклад, для заповнення сектора на диску).

Файли і функції файлової системи у мові С

У мові С файлом може бути все що завгодно, починаючи з дискового файлу і закінчуючи терміналом або принтером. Потік пов'язують з певним файлом, виконуючи операцію відкриття.

Як тільки файл відкритий, можна проводити обмін інформацією між ним і програмою. При відкритті файлу покажчик поточної позиції у файлі встановлюється в початок. При читанні з файлу (або записі в нього)

кожного символу покажчик поточної позиції збільшується, забезпечуючи просування по файлу.

Файл від'єднується від певного потоку (тобто розривається зв'язок між файлом і потоком) за допомогою операції закриття.

У кожного потоку, пов'язаного з файлом, є керуюча структура, яка містить інформацію про файл, вона має тип *FILE*. У цьому блоці управління файлом ніколи нічого міняти не можна.

Для роботи з файловою системою існує заголовочний файл <*stdio.h*>. Часто використовувані функції файлової системи С такі (табл. 8.1).

Таблиця 8.1 – Функції для роботи з файловою системою

Назва	Що робить
fopen()	Відкриває файл
fclose()	Закриває файл
putc()	Записує символ у файл
fputc()	Те саме, що і putc()
getc()	Читає символ з файлу
fgetc()	Те саме, що і getc()
fgets()	Читає рядок з файлу
fputs()	Записує рядок у файл
fseek()	Встановлює покажчик поточної позиції на певний байт файлу
ftell()	Повертає поточне значення покажчина у файлі
fprintf()	Для файлу те саме, що printf() для консолі
fscanf()	Для файлу те саме, що scanf() для консолі
feof()	Повертає значення true (істина), якщо досягнуто кінець файлу
error()	Повертає значення true, якщо виникла помилка
rewind()	Встановлює покажчик поточної позиції на початок файлу
remove()	Знищує файл
fflush()	Дозапис потоку у файл

Заголовок <*stdio.h*> надає прототипи функцій введення/виведення і визначає наступні три типи: *size_t, fpos_t* і *FILE*. *size_t* і *fpos_t* представляють собою певні різновиди такого типу, як ціле без знака. А про третій тип, *FILE*, розповідається в наступному розділі.

Крім того, в <*stdio.h*> визначається декілька макросів. Серед них:

NULL, EOF, FOPEN_MAX, SEEK_SET, SEEK_CUR і *SEEK_END.*

Макрос *NULL* визначає порожній (*null*) покажчик. Макрос *EOF* часто визначається як -1, і є значенням, що повертається тоді, коли функція введення намагається виконати читання після кінця файлу. *FOPEN_MAX* – ціле значення, рівне максимальному числу одночасно відкритих файлів.

Інші макроси вико ристовуються разом з *fseek()*, що виконує операції прямого доступу до файлу.

Щоб оголосити змінну-покажчик файлу, пишуть:

```
FILE *fp;
```

Функція ***fopen()*** відкриває потік і пов'язує з цим потоком певний файл. Потім вона повертає покажчик цього файлу:

```
FILE *fopen (const char *ім'я_файлу,const char *режим);
```

Рядок "`режим`", визначає, яким чином файл буде відкритий. Рядки, подібні "*r+b*" можуть бути представлені і у вигляді "*rb+*" (табл. 8.2).

Таблия 8.2 – Режими відкриття файлів

Режим	Що означає
r	Відкрити текстовий файл для читення
w	Створити текстовий файл для запису
a	Додати в кінець текстового файлу. Якщо файл не існує, то він буде створений. Всі нові дані, які записуються в нього, будуть додаватися в кінець файлу.
rb	Відкрити двійковий файл для читання
wb	Створити двійковий файл для запису
ab	Додати в кінець двійкового файла
r+	Відкрити текстовий файл для читання/запису. Вміст залишиться недоторканим. Якщо файлу не існує, то він створений не буде.
w+	Створити текстовий файл для читання /запису. Якщо файл не існує, то він буде створений. Якщо файл вже існує, то відкриття призведе до втрати його вмісту, а в режимі r+ він залишиться недоторканим
a+	Додати в кінець текстового файлу або створити його для читання/запису
r+b	Відкрити двійковий файл для читання /запису
w+b	Створити двійковий файл для читання /запису
a+b	Додати в кінець двійкового файла або створити його для читання /запису

Робота з файловими потоками

У мові С++ введення/виведення описується як набір класів, описаний в заголовному файлі *iostream.h.* Аналогами потоків *stdin, stdout, stderr* є класи *cin, cout* і *cerr*. Ці три потоки відкриваються автоматично. Потік *cin* пов'язаний з клавіатурою, а *cout, cerr* – з дисплеєм.

Файл <*fstream.h*> визначає класи *ifstream* і *ofstream*, за допомогою яких програма може виконувати операції файлового введення/виведення. Для відкриття файлу на введення/виведення оголошують об'єкт типу *ifstream/ ofstream*, передаючи конструктору цього об'єкта ім'я необхідного файлу:

```
ofstream myOutput ("FileOut.EXT");
ifstream myInput    ("FileIn.EXT");
```

Після того, як програма відкрила файл для введення або виведення, вона може читати або писати дані, використовуючи оператори: "<<" – для занесення (запису) в потік; ">>" – для вилучення (читання) з потоку.

```
char word [64];
while (!myInput.eof ())
{     myInput >> word;          // зчитуэмо слово (до пробілу)
cout <<word <<endl;      // виводимо на екран
```

```
        myOutput << word; // записуємо у файл
}
```

Більшість програм читають вміст файлу, поки не зустрінеться *кінець файлу*. Визначити кінець файлу можна за допомогою функції ***eof()*.**

Для введення або виведення *символів* у файл або з файлу використовую функції ***get()*** і ***put()***.

```
char letter;
while (!myInput.eof ())
{     letter = myInput.get(); // зчитуємо з файлу
      cout <<letter;                // виводимо на екран
      myOutput.put(letter);         // записуємо у файл
}
```

Для зчитування *цілого рядка* використовують функцію ***getline()***:

```
char line [80];
while (!myInput.eof ())
{     myInput.getline (line,sizeof (line));
cout <<line <<endl;
}
```

Для перевірки помилок можна використовувати функцію ***fail()***:

```
if (myInput.fail ())
{     cerr <<"Помилка відкриття FileIn.txt" <<endl;
      exit (1);
}
```

Якщо програмам необхідно вводити або виводити такі дані, як структури або масиви, можна використовувати методи ***read()*** і ***write()***.

```
myInput.read (buffer, sizeof (buffer));
myOutput.write (buffer, sizeof (buffer));
```

Якщо програма завершила роботу з файлом, його слід закрити за допомогою функції ***close()*.**

Для того, щоб операції введення/виведення виконувалися не з початку файлу, можна використовувати інші режими відкриття файлів (табл. 8.3).

Таблиця 8.3 – Режими відкриття файлових потоків

Режим відкриття	Призначення
ios::app	Відкриває файл в режимі додавання, встановлюючи файловий покажчик на кінець файлу
ios::ate	Встановлює файловий покажчик на кінець файлу
ios::in	Вказує відкрити файл для введення .
ios::nocreate	Якщо й файл не існує, не створювати файл і повернути помилку
ios::noreplace	Якщо файл існує, операція відкриття повинна бути перервана и повинна повернути помилку
ios::out	Вказує відкрити файл для виведення .
ios::trunc	Перезаписує вміст існуючого файлу
ios::binary	Робота з файлом у двійковому вигляді

Наприклад,

```
ifstream myFile ("Filename.txt", ios::out | ios::noreplace);
```

Для читання і запису даних будь-якого типу, тип яких може займати більше 1 байта, у файловій системі мови С є дві функції: ***fread*()** і ***fwrite*().**

```
size_t fread(void *buf,size_t count,size_t k,FILE *pf);
```

```
size_t fwrite(const void *buf,size_t count,size_t k,FILE *pf);
```

Функція *fread*() повертає кількість прочитаних елементів. Якщо досягнуто кінець файлу або сталася помилка, то повернуте значення може бути менше, ніж лічильник. А функція *fwrite*() повертає кількість записаних елементів. Якщо помилки не було, то повернений результат буде дорівнює значенню лічильник. Одним з найбільш корисних застосувань функцій *fread*() і *fwrite*() є читання і запис даних користувача типів. Наприклад, якщо визначена структура

```
struct struct_type {
      float balance;
      char name [80];
} Cust;
```

то наступний оператор записує вміст *Cust* у файл, на який вказує *fp*:

```
fwrite (&cust, sizeof (struct struct_type), 1, fp);
```

В системі введення/виведення мови С є функції ***fprintf()*** і ***fscanf()***:

```
int  fprintf (FILE * pf, const char *str, ...);
int  fscanf (FILE * pf, const char *str, ...);
```

Приклади програм для робот из файлами

Приклад 1. Введення інформації з клавіатури і виведення на диск.

```
int main (int argc, char * argv [])
{
      FILE * fp;
      char ch;
      if (argc! = 2)
      {
            printf ("Ви забули ввести ім'я файлу. \n");
            exit (1);
      }
      if ((fp = fopen (argv [1], "w "))== NULL)
      {
            printf ("Помилка при відкритті файлу. \n");
            exit (1);
      }
      do {  ch = getchar ();
            putc (ch, fp);
      } while (ch! ='$');
      fclose (fp);
      return 0;
 }
```

Приклад 2. Програма читає текстовий файл і виводить його на екран

```
int main (int argc, char * argv [])
{
      FILE * fp;
      char ch;
      if (argc! = 2) {
            printf ("Ви забули ввести ім'я файлу. \n");
            exit (1);
      }
      if ((fp = fopen (argv [1], "r "))== NULL) {
            printf ("Помилка при відкритті файлу. \n");
```

```
            exit (1);
        }
        ch = getc (fp); /* читання одного символу */
        while (ch! = EOF) {
            putchar (ch); /* виведення на екран */
            ch = getc (fp);
        }
        fclose (fp);
    return 0;
}
```

Приклад 3. Програма вводить рядок з клавіатури, записує у файл, а потім читає файл і виводить його вміст на екран

```
int main (void)
{   char str[80];
    FILE *fp;
    if ((fp = fopen ("TEST", "w+"))== NULL) {
        printf ("Помилка при відкритті файлу. \n");
    exit (1);
    }
    do {  printf("Введіть рядок (порожній – для виходу): \n");
        gets(str);
        strcat(str, "\n"); /* введення роздільник рядків */
        fputs(str, fp);
    } while(*str != '\n');
        /* Тепер виконується читання і відображення файлу */
    rewind(fp); /* встановити покажчик  на початок файлу */
    while (!feof(fp))
    {   fgets (str,79,fp);
        printf (str);
    }
    return 0;
}
```

Приклад 4. Використання форматованого введення-виведення у файл

```
int main (void)
{
    FILE *fp;
    char s[80];
    int t;
    if ((fp = fopen ("test", "w")) == NULL)
    {   printf ("Помилка відкриття файлу. \n");
        exit (1);
    }
    printf ("Введіть рядок і число:");
    fscanf (stdin, "%s%d",s,&t);   // читати з клавіатури
    fprintf (fp, "%s%d",s,t);           // писати в файл
    fclose (fp);
    if ((fp = fopen ("test", "r")) == NULL)
    {   printf ("Помилка при відкритті файлу. \n");
        exit (1);
    }
    fscanf (fp, "%s%d",s,&t);           // читання з файлу
    fprintf (stdout, "%s%d",s,t);  // виведення на екран
    return 0;
}
```

Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом, з функціями стандартних бібліотек для роботи файлами і файловими потоками.
2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програму на своєму комп'ютері.
3. Розробити власні програми, які реалізує індивідуальне завдання.
4. Підготувати звіт по кожній задачі:
   – варіант і текст завдання;
   – лістинг програми;
   – схему даних (для обох задач) і схему взаємозв'язку функцій (лише для другоі задачі);
   – роздруківку вхідних і вихідних файлів і результати виконання;
   – висновки.

Варіанти індивідуальних завдань

Задача 1. Взявши за основу *лабораторну роботу №5*, змінити код програми таким чином, щоб:
– вхідні дані вводилися не з клавіатури, а з файлу,
– результати виконання виводились і на екран, і у файл.

Задача 2. Взявши за основу *лабораторну роботу №7(задача 1)*, змінити код програми таким чином, щоб:
– вхідні дані (поля структури) вводилися з клавіатури і після введення записувалися у файл (окрема функція);
– програма мала можливість дописувати дані у файл (окрема функція);
– дані з файлу виводились на екран (окрема функція);
– результати виконання другого підпункту виводились на екран і у файл.
При цьому виконати завдання задачі №2 у двох варіантах:
1) за допомогою класів потоків *ofstream* та *ifstream*;
2) за допомогою структури *FILE* та функцій роботи з нею.

***** ***Задача 4.*** Одним за найпростіших в реалізації та найшвидших методів шифрування є накладання гами – послідовності псевдовипадкових чисел – шляхом виконання операції побітового додавання за модулем 2 (також відомої як exclusive or чи скорочено XOR).

Розробіть програму для зашифрування та розшифрування файлів з довільним розширенням, використовуючи як ключ, на основі якого генерується послідовність псевдовипадкових чисел, номер варіанту.

Контрольні питання

1. Дати поняття потоків і файлів. Який між ними зв’язок?
2. Які види потоків бувають? Пояснити різницю між ними.
3. Охарактеризувати стандартні потоки введення-виведення.
4. Яким чином можна оголосити і відкрити файл?
5. Шо таке режими відкриття файлів і які режими ви знаєте?
6. Наведіть приклади форматованого запису у файл і читання з нього.
7. Яким чином можна встановлювати покажчик файлу у задану позицію?
8. Яким чином можна здійснювати запис і зчитування блоками?
9. Чи можна одночасно використовувати файл для запису і читання?
10. Що таке файлові потоки? Які поняття програмування забезпечують роботу з ними? Назвіть їх.
11. Основні функції роботи з файловими потоками.
12. Які режими доступу до файлових потоків ви знаєте?
13. Яким чином можна виводити у файлові потоки і вводити великі обсяги даних (структури, масиви)?
14. Як можна перевірити, чи закінчився файл, і наявність помилок?
15. Наведіть приклади запису і читання по символах, по словах, по рядках та блоками довільного розміру.

ЛАБОРАТОРНА РОБОТА №9

КОНСТРУЮВАННЯ БАГАТОФАЙЛОВИХ ПРОГРАМ

Мета роботи

- Вивчення основних директив препроцесорної обробки програм.
- Отримання практичних навиків у роботі зі структурами, функціями і модульною структурою програм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Директиви препроцесорної обробки

Директиви – інструкції препроцесора. Обробка програми препроцесором здійснюється перед її компіляцією. Призначення препроцесорної обробки:

- включення у файл, що компілюється, інших файлів,
- визначення символічних констант і макросів,
- встановлення режиму умовної компіляції програми і умовного виконання директив препроцесора.

Синтаксис:

- директиви повинні починатися з символу "#".
- після директив препроцесора не ставиться " ;".

Основні директиви:

#include **–** включає копії вказаного файла в те місце програми, де знаходиться ця директива;

#define **–** здійснює **макропідстановку** – заміняє всі входження ідентифікатора у програмі на текст, що слідує в директиві за ідентифікатором;

#undef **–** анулює визначення символічних констант і макросів;

#if, #elif, #else, #endif **–** здійснюють умовну компіляцію;

#ifdef і ***#ifndef*** **–** використовуються замість *#if – defined* (…);

#line **–** повідомляє компілятору про зміну імені програми і порядку нумерації рядків;

#error **–** вказує компілятору, що треба надрукувати повідомлення про помилку і перервати компіляцію (зазвичай використовують у середині умовних директив);

#pragma **–** дозволяє керувати можливостями компіляції.

Існує і ряд інших директив.

Визначені макроси ANSI:

__DATE__ **–** рядок у формі mmm.dd.yyyy – дат створення даного файлу;

__TIME__ **–** час початку обробки поточного файлу у форматі hh:mm:ss ;

__FILE__ – ім'я поточного файлу;
__LINE__ – номер поточного рядка;
__STDC__ – визначений, якщо встановлено режим сумісності з ANSI.

Багатофайлові проекти

У мові С модулі на рівні синтаксису не виділяються. Реалізація модулів можлива з використанням роздільної компіляції та файлів заголовків. Програма на С може складатись з декількох файлів, кожний з яких вміщує підпрограми та описи. Головним файлом вважається той, що містить функцію *main()*.

Роздільна компіляція – це обробка компілятором кожного файлу програми окремо. Файли заголовків мають розширення "*.h*" та вміщують описи глобальних констант, типів, змінних, підпрограм. Описи констант, типів, змінних не відрізняються від розглянутих вище. Для підпрограм у файлах заголовків наводять тільки заголовок, після якого ставлять ";":

```
extern t f(t1 x1, ..., tn xn);
```

де t, t_i – тип, x_i – змінні.

Ключове слово *extern* вказує на те, що підпрограма реалізована у зовнішньому файлі.

Файли заголовків підключають до інших файлів за допомогою директиви *#include*. Наприклад,

```
#include "myfile.h"
```

Ми вже знайомі з цією директивою та постійно користувались нею для підключення стандартних бібліотек. Різниця із підключенням власних файлів заголовків в тому, що для стандартних бібліотек вико ристовують символи *"< >"* замість лапок:

```
#include <stdio.h>
```

При реалізації модулів для кожного модуля у С створюють два файли з однаковим іменем та з розширеннями "*.h"* та "*.cpp"*. Файл заголовку з розширенням *.h* грає роль інтерфейсної частини модуля, а файл з розширенням "*.cpp"* – частини реалізації модуля. Для використання модуля треба підключити відповідний файл заголовка за допомогою *#include*.

Приклад розробки багатофайлового проекту

Задача. Необхідно забезпечити роботу з таблицею, яка містить інформацію про студентів.

Формалізація задачі

1. Програма повинна складатися, як мінімум, з двох файлів:
   а) у першому повинна знаходитися головна програма, функції якої – вибір у діалоговому режимі однієї з вищевказаних дій;
   б) у інших – функції, які безпосередньо реалізують ці дії.

2. Визначаємо змінні, загальні для усіх модулів програми.
   По-перше, необхідно оголосити структуру, яка являє собою рядок таблиці:

```
struct stud {
```

```
        char name[25]; // Ф.І.Пб.
        chat gend;     // стать
        int year;      // рік народження
        float sq;      // вага
}
```

Оскільки необхідно створити програмний засіб з декількох файлів, виділимо це оголошення в окремий файл-заголовок, який буде включатися (`#include`) у всі файли-програми. Це забезпечить однаковість цієї структури в усіх модулях програми. Для скорочення запису визначимо скорочені імена для структури і її розміру:

```
#define STUD struct stud
#define SSTUD sizeof(struct stud)
```

Дані таблиці будуть знаходитися в масиві `STUD st[N]`.

Розмірність масиву визначимо через макроконстанту:

```
#define N 100
```

Окрім того, що цей масив повинен розміщуватись в оперативній пам'яті, він ще повинен бути доступним для декількох модулів програми. Тобто, в одному модулі він повинен бути оголошений поза блоками, а в інших (які до нього звертаються) – описаний як зовнішній.

```
extern STUD st[];
```

Те саме стосується і змінної, яка міститиме кількість заповнених елементів у масиві. Вона повинна бути визначена в одному модулі:

```
int n;
```

а в інших описана як зовнішня:

```
extern int n.
```

3. Модулі програми

Окремий модуль програми буде складати головна функція програми (це загальноприйнятий підхід). В ньому розміщається програма-монітор, яка керує порядком виконання інших функцій програмного засобу (він буде називатись `mainProg`).

Розробимо ще два модуля:

- один такий, що вимагає звернення до глобальних змінних програми;
- другий такий, що його функції не залежать від глобальних змінних.

Для кожного з модулів зробимо ще і заголовочний файл. Це модулі `Prog1` і `Prog2`.

3.1 Розробка модуля `mainProg`

Даний модуль міститиме головну функцію, яка керуватиме порядком виконання інших функцій програмного засобу. В цьому модулі необхідно зібрати глобальні змінні програми і функцію `main`(). Локальні змінні функції `main`():

`op` – код тієї дії, яку повинна виконати програма;

`num` – номер запису для тих дій, які його потребують;

`eoj` – ознака кінця роботи.

3.2 Розробка модуля `Prog1`

У файл `Prog1.h` поміщаємо усі прототипи функцій, а у файлі `Prog1.cpp` – опис цих функцій. При цьому підключаємо потрібні заголовочні файли.

Тут зібрані функції, які мають доступ до глобальних змінних – масиву і кількості змінних у ньому. Тому саме тут міститься оголошення зовнішніх змінних, а також визначення функцій:

`check_number()` – перевірка правильності номеру елемента (0 – вірний, 1 – ні),

`del_item()` – видалення елементу із заданим номером з масиву,

`show_all()` – виведення на екран усього масиву. Ця функція викликатиме інші 3 функції, які не залежать від глобальних змінних і будуть описані у модулі `Prog2`: `print_head()`,`show_row()`,`print_line()`.

3.3 Розробка модуля `Prog2`

Тут визначені функції, які не залежать від загальних змінних.

`get_number()` – введення з клавіатури номера елемента масиву;

`enter_data()` – введення з клавіатури значень для одного елемента масиву;

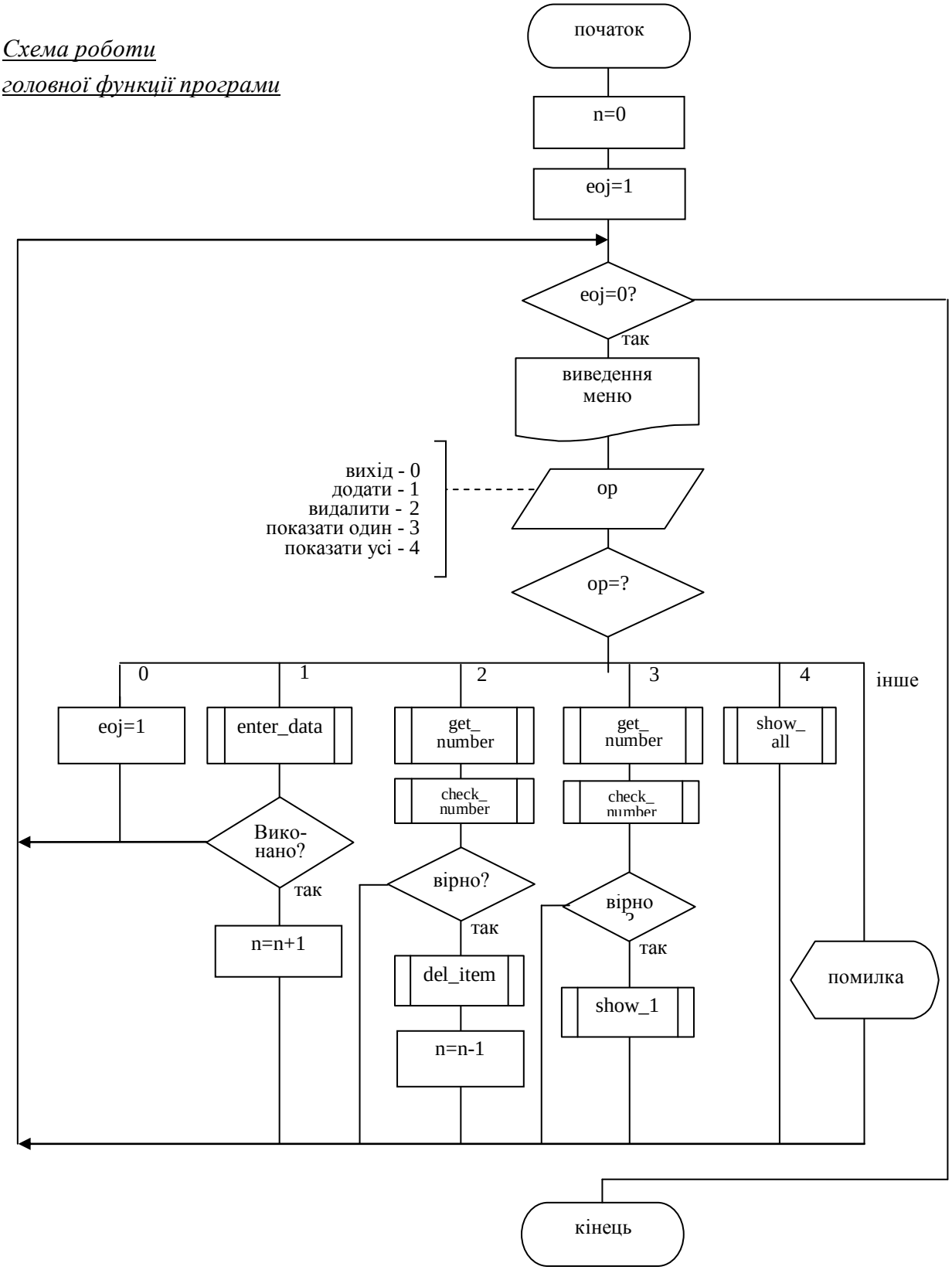
`show_1()` – виведення на екран одного елемента масиву;

`print_head()` – виведення заголовка таблиці;

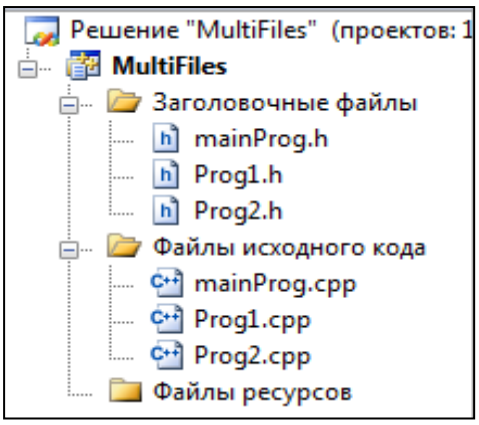
`show_row()` – виведення одного рядка таблиці;

`print_line()` – виведення нижньої лінії таблиці.

Схема роботи головної функції програми



Створення програмного проекту



Проект, що реалізує програмний засіб, буде складатися з шести файлів.

Файл `mainProg.h`

```
#define STUD struct stud
#define SSTUD sizeof(struct stud)
struct stud {
  char name[25];     // Ф.І.По.
  char gend;         // стать
  int year;          // рік народження
  float w;  };       // вага
```

Файл `mainProg.cpp`:

```
#include "mainProg.h"
#include "Prog1.h"
#include "Prog2.h"
#define N 100
STUD stud[N]; /* масив-таблиця */
int n=0;    /* кількість елементів у масиві */
int main(void) {
  int op;   /* операція */
  int num;  /* номер елементу */
  char eoj; /* познака кінця */
  setlocale(0,"");
  for (eoj=0; !eoj; ) {/* виводення меню */
      printf("1 - Додати елемент\n");
      printf("2 - Видалити елемент\n");
      printf("3 - Показать елемент за номером\n");
      printf("4 - Показати все\n");
      printf("0 - Вихід\n");
      printf("Введіть >  ");
      scanf("%d",&op); /* вибір з меню  */
      switch(op) {
        case 0: eoj=1;break; /* вихід */
        case 1: if (!enter_data(stud+n)) n++;/* додати */
                break;
        case 2: if (!check_number(num=get_number())) {/* видалити*/
                    del_item(num);
                    n--;
                } break;
        case 3: if (!check_number(num=get_number()))/* показати один */
                    show_1(stud+num-1);
                break;
        case 4: show_all();/* показать все */
                break;
        default: printf("Неправильна операція\n");
                 break;
        }  /* switch */
      if (op) {
        printf("Нажмите любую клавишу\n");
        getch();
        } /* if */
      } /* for */
      return 0;
  }  /* main */
```

Файл Prog1.h

```
int check_number(int);
void del_item(int);
void show_all(void);
```

Файл Prog1.cpp

```
#include <stdio.h>
#include <memory.h>
#include "mainProg.h"
#include "Prog2.h"
extern STUD stud[];
extern int n;

int check_number(int a) {/**** перевірка номера елемента ****/
  if (a<1)     { printf("Минимальный номер : 1\n");   return -1;      }
  if (a>n)  {  printf("Максимальный номер : %d\n",n);    return -1;    }
  return 0;
}

void del_item(int m) {/**** видалення елемента ****/
  int i;
  for (; m<n; m++) memcpy(stud+m-1,stud+m,SSTUD);
}

void show_all() {/**** вывод всего массива ****/
  int i;
  print_head();
  for (i=0; i<n; i++) show_row(stud+i);
  print_line();
}
```

Файл Prog2.h

```
void print_head(void);
void print_line(void);
void show_row(STUD *);
int get_number(void);
void show_1(STUD *);
int enter_data(STUD *);
```

Файл Prog2.cpp:

```
#include <stdio.h>
#include <string.h>
#include "mainProg.h"
#include "Prog2.h"

int get_number() {/****  введення номера ****/
  int b;
  printf("Введите номер>");
  scanf("%d",&b);
  return b;
}

int enter_data(STUD *s) {/**** введення даних про одного студента ***/
  float f;
        printf("Введите имя, пол, год рождения, вес:\n");
  scanf("%s %c %d %f",s->name,&s->gend,&s->year,&f);
  s->w=f;
  if (!strcmp(s->name,"***")) return -1;
```

```
  return 0;
}
void show_1(STUD *s) {/**** виведення даних про одного студента ***/
  printf("\nП.І.Б.       : %s\n",s->name);
  printf("Стать          : %c");
  switch(s->gend)
  {
        case 'м': printf("(чол.)"); break;
        case 'ж': printf("(жін.)"); break;
  }
  printf("\nРік народження : %d\n",s->year);
  printf("Вага            : %6.2f\n",s->w);
  print_line();
}

void show_row(STUD *s) {/**** виведення рядка таблиці ****/
  printf("| %9s |   %c   |       %3d | %-5.1f       |\n",
    s->name,s->gend,s->year,s->w);
}

void print_line()
{
  printf("-----------------------------------------------\n");
}

void print_head()
{
  print_line();
  printf("|              СПИСОК СТУДЕНТІВ                |\n");
  printf("|----------------------------------------------|\n");
  printf("|     Ім'я  | Стать |    Рік    |    Вага     |\n");
  printf("|           |       |народження |             |\n");
  printf("|-----------|-------|-----------|-------------|\n");
}
```

Результати роботи програми (фрагмент)

```
1 - Додати елемент
2 - Видалити елемент
3 - Показать елемент за номером
4 - Показати все
0 - Вихід
Введіть >  1
Введите имя, пол, год рождения, вес:
Sidorov m 1992 65


Нажмите любую клавишу
1 - Додати елемент
2 - Видалити елемент
3 - Показать елемент за номером
4 - Показати все
0 - Вихід
Введіть >  4
-----------------------------------------------
|              СПИСОК СТУДЕНТІВ                |
|----------------------------------------------|
|     ?м'я  | Стать |    Рік    |    Вага     |
|           |       |народження |             |
|-----------|-------|-----------|-------------|
|   Petrova |   w   |       1990 | 58,0       |
|   Sidorov |   m   |       1992 | 65,0       |
-----------------------------------------------


Нажмите любую клавишу
```

Порядок виконання роботи

1. За результатами виконання *лабораторних робіт №№7-8* розробити у вигляді багатофайлового проекту програму, яка буде відповідати таким вимогам:
   а) заповнення масиву (таблиці) даних повинно здійснюватись із зовнішнього файлу;
   б) програма повинна забезпечувати роботу з даними у різних режимах:
   – додавання нових рядків у таблицю;
   – видалення рядка із заданим номером з таблиці;
   – виведення інформації, яка зберігається у рядку с заданим номером;
   – виведення на екран усієї таблиці.
2. Підготувати звіт:
   – варіант і текст завдання;
   – загальна схема функціонування програми;
   – схема взаємозв’язків функцій у програмі (з нанесенням на схему вхідних і вихідних даних для кожної з функцій;
   – лістинг програми (по файлах);
   – результати виконання з відображенням результатів роботи кожного режиму;
   – висновки.

Контрольні питання

1. Що таке директиви препроцесора?
2. Який синтаксис написання директив препроцесора?
3. Коли здійснюється обробка програми препроцессором?
4. Наведіть основні директиви препроцессора і поясніть їх дію.
5. Що таке "препроцесорна обгортка"? У яких випадках її використовують?
6. Що таке макровизначення? Як вони працюють?
7. У чому переваги використання багатофайлових програм?
8. Які класи пам’яті ви знаєте?
9. У яких випадках використовують регістровий тип даних?
10. Що означають локальні змінні? До якого класу пам’яті вони належать?
11. До якого класу пам’яті можна віднести формальні параметри функцій?
12. Як і для чого використовують зовнішні глобальні змінні?компіляція
13. Що таке роздільна компіляція?
14. Що рекомендується зберігати у заголовочних файлах, а що – у програмних файлах?

ЛАБОРАТОРНА РОБОТА №10

СОРТУВАННЯ І ПОШУК

Мета роботи

- Отримання практичних навичок в обробці масивів, у сортуванні елементів масивів різними методами та за різними реквізитами.
- Дослідження і вивчення методів пошуку ключових елементів у масивах.
- Здійснення порівняння та аналізу ефективності використовуваних методів сортування і пошуку.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Бульбашкове сортування (сортування обміном)

Метод "бульбашкового сортування" ґрунтується на перестановці сусідніх елементів. Для впорядкування елементів масиву здійснюються повторні проходи по масиву. Переміщення елементів масиву здійснюється таким чином: масив переглядається зліва направо, здійснюється порівняння пари сусідніх елементів; якщо елементи в парі розміщені в порядку зростання, вони лишаються без змін, а якщо ні – міняються місцями. В результаті 1-го проходу найбільше число буде поставлено в кінець масиву. У 2-му проході операції виконуються над елементами з першого до (N-1)-ого, у 3-му – від першого до (N-2)-ого і т.д. Впорядкування масиву буде закінчено, якщо при проході масиву не виконається жодної перестановки елементів масиву. Факт перестановки фіксується за допомогою деякої змінної (*is*), яка на початку має значення 0 і набуває значення 1, коли виконається перестановка в будь-якій парі.

Масив до впорядкування	22	20	-1	-40	88	-75	-22
Перший перегляд масиву	20	-1	-40	22	-75	-22	**88**
Другий перегляд масиву	-1	-40	20	-75	-22	**22**	**88**
Третій перегляд масиву	-40	-1	-75	-22	**20**	**22**	**88**
Четвертий перегляд масиву	-40	-75	-22	**-1**	**20**	**22**	**88**
П'ятий перегляд масиву	-75	-40	**-22**	**-1**	**20**	**22**	**88**

```
const n=10;
int a[n], i, c, is;
/* … */
do {  is=0;
      for (i=1;i<n;i++)
      if (a[i-1]>a[i])
      {     c=a[i]; a[i]=a[i-1]; a[i-1]=c;
            is=1;
      }
} while (is);
```

Сортування методом вибору

Сутність методу така: масив переглядається перший раз, знаходиться мінімальний елемент масиву і міняється місцями з першим елементом. Другий раз масив переглядається, починаючи з другого елементу. Знову знаходиться мінімальний елемент і міняється місцями з другим. Даний процес виконується доти, поки не буде поставлений на місце *N-1* елемент.

Масив до впорядкування	22	20	-1	-40	88	-75	-22
Перший перегляд масиву	**-75**	20	-1	-40	88	**22**	-22
Другий перегляд масиву	-75	**-40**	-1	**20**	88	22	-22
Третій перегляд масиву	-75	-40	**-22**	20	88	22	**-1**
Четвертий перегляд масиву	-75	-40	-22	**-1**	88	22	**20**
П'ятий перегляд масиву	-75	-40	-22	-1	**20**	22	**88**
Шостий перегляд масиву	-75	-40	-22	-1	20	22	88

```
const int n=20;
int b[n];
int imin, i, j, a;
/* … */
for (i=0;i<n-1;i++)
{    imin=i;
     for (j=i+1;j<n;j++)
          if (b[j]<b[imin]) imin=j;
     a=b[i];
     b[i]=b[imin];
     b[imin]=a;
}
```

Сортування вставками

При використанні даного методу на *i*-му етапі відбувається "вставка" елемента *a[i]* в потрібну позицію серед елементів *a[1], ..., a[i-1]*, які вже впорядковані. Після цієї вставки перші *i* елементів будуть впорядковані.

Масив до впорядкування	22	20	-1	-40	88	-75	-22
Перший перегляд масиву	20	22	-1	-40	88	-75	-22
Другий перегляд масиву	-1	20	22	-40	88	-75	-22
Третій перегляд масиву	-40	-1	20	22	88	-75	-22
Четвертий перегляд масиву	-40	-1	20	22	88	-75	-22
П'ятий перегляд масиву	-75	-40	-1	20	22	88	-22
Шостий перегляд масиву	-75	-40	-22	-1	20	22	88

```
const int n=20;
int b[n];
int i,j,c;
/* … */
for (i=1;i<n;i++)
{    c=a[i];
     for (j=i-1;j>=0&&a[j]>c;j--)
          a[j+1]=a[j];
     a[j+1]=c;
}
```

Швидке сортування

Швидке сортування полягає в тому, що множина елементів B $\{k_1, ..., k_n\}$ перетворюється на множину B_1, $\{k_1\}$, B_2, де B_1 – підмножина B з елементами, не більшими за k_1, а B_2 – підмножина B з елементами більшими k_1. Причому k_1 після розбиття множини B буде перебувати на потрібному місці. Далі до множин B_1 і B_2 знову застосовують впорядкування швидким сортуванням. На практиці цей алгоритм виявляється одним із найшвидших.

```
double * quick(double *s,int low,int hi)
{ double cnt,aux;
  int i,j;
  if (hi>low)
  {    i=low; j=hi;
       cnt=s[i];
       while(i < j)
       {     if (s[i+1]<=cnt)
             {     s[i]=s[i+1];
                   s[i+1]=cnt;
                   i++;
             }
             else
             {     if (s[j]<=cnt)
                   {     aux=s[j];
                         s[j]=s[i+1];
                         s[i+1]=aux;
                   }
                   j--;
             }
       }
       quick(s,low,i-1);
       quick(s,i+1,hi);
  }
  return(s);
}
```

Сортування методом Шелла

Головний недолік простих методів – обмін ведеться в основному між сусідними елементами. Тому бажано робити якомога ширші обміни.

Метод Шелла – метод сортування включеннями з відстанями, що зменшуються. Візьмемо для прикладу масив:

60 16 41 06 59 79 34 15

На першому етапі масив уявно ділиться на підмасиви з елементами, що, скажімо, відстоять один від одного на 4 елементи:

60 59
16 79
41 34
06 15

Кожен з них впорядковується окремо:

59 60
16 79

34 41
06 15

Сортування робиться у кожному підмасиві методом простого включення. На другому етапі підмасиви утворюються елементами через один:

59 34 60 41
16 06 79 15

Одержуємо:

34 41 59 60
06 15 16 79

Після цього весь масив сортується разом. За рахунок попередніх етапів він виявляється вже близьким до відсортованого, тому обмінів необхідно вже не так багато. Цей метод дає кращі результати, якщо розподіл на підмасиви роблять кроками, що не є степенями двійки, а, навпаки, не є множниками один одного.

```
void SortShell(int* arr, int n)
{
  int step = n / 2; // не обов'язково!
  while (step > 0)
  {
    for(int i = 0; i < n-step; i++)
    {
      int j=i;
      while (j>=0 && a[j]>s[j+step])
           {   int t=a[j];
               a[j]=a[j+step];
               a[j+step]=t;
               j--;
           }
      }
    step /= 2;
  }
}
```

Лінійний пошук

Лінійний пошук порівнює кожен елемент масиву з ключем пошуку. Оскільки масив не впорядкований, цілком імовірно, що відшукують значення виявиться першим елементом масиву. Але в середньому програма повинна порівняти з ключем пошуку половину елементів масиву.

```
int LinearSearch (int array [], int size, int key)
{
    for (int i = 0; i <size; i + +)
    if (array [i] == key)
    return i;
    return -1;
}
```

Метод лінійного пошуку добре працює для невеликих або для несортованих масивів. І хоча для великих масивів лінійний пошук неефективний, він абсолютно надійний.

Двійковий пошук

Двійковий (бінарний) пошук – алгоритм пошуку елемента у відсортованому масиві. Бінарний пошук знайшов собі застосування в математиці і інформатиці. Двійковий пошук можна використовувати тільки в тому випадку, якщо є масив, всі елементи якого впорядковані (відсортовані).

Нехай *x* – елемент, пошук якого здійснюється; *l (left)* – ліва границя пошуку; *r (right)* – права границя пошуку.

```
int binary_search(int x, int l ,int r)
{
     while(r – l > 1)
     {
           int mid = (l + r) / 2;
                //ділимо відрізок [l,r] навпіл
           if(a[mid]<x)     l = mid;
            else  r = mid;
    }
     for(int i = l; i <= r; i++)
         if(a[i]==x)
           return i;
     return -1;// не знайшли такого элемента
}
```

Порядок виконання роботи

1. Засвоїти теоретичний матеріал. Розібратись у сутності кожного методу сортування та пошуку.
2. Розробити програму згідно індивідуального завдання. Для сортування і пошуку повинні бути розроблені окремі функції.
3. Підготувати звіт:
   - варіант і текст завдання;
   - схеми роботи функцій сортування і пошуку;
   - лістинг програми;
   - результати виконання і аналізу сортування (вигляди екрану, таблиця і графік);
   - висновки.

Варіанти індивідуальних завдань

1. Сгенерувати масив (масив оголошувати динамічно) і вивести його на екран (використати при цьому власну функцію для виведення масиву на екран).
2. Відсортувати масив, обравши вказані методи та параметри сортування.
3. Запускаючи програму не менше десяти разів для кожного методу (або передбачити це у програмі), задавати різну кількість *n* елементів маси-

ву, отримати час *t* сортування. Побудувати залежність *t=f(n)* на одному графіку для різних методів сортування (у табличному і графічному виглядах).

4. Здійснити пошук вказаного (з клавіатури) елемента у масиві, використовуючи вказаний метод пошуку).

№	Методи сортування	Тип елементів масиву	Напрям сортування	Порядок сортування	Метод пошуку
1	Вибору Швидкий	Ціле	З початку	За убуванням	Бінарний
2	Вибору Вставки	Символьне	З початку	За зростанням	Лінійний
3	Шелла. Вибору	Дійсне	З кінця	За убуванням	Бінарний
4	Вставки. Бульбашковий	Довге ціле	З кінця	За зростанням	Бінарний
5	Шейкерний. Вибору	Символьне	З початку	За убуванням	Лінійний
6	Швидкий. Бульбашковий	Дійсне (подв. точність)	З початку	За зростанням	Лінійний
7	Вставки. Шейкерний	Коротке ціле	З кінця	За убуванням	Бінарний
8	Шелла. Шейкерний	Символьне	З кінця	За зростанням	Бінарний
9	Бульбашковий. Вибору	Дійсне	З початку	За убуванням	Лінійний
10	Вставки. Шелла	Ціле	З початку	За зростанням	Бінарний
11	Шейкерний. Швидкий	Символьне	З кінця	За убуванням	Бінарний
12	Вибору. Шейкерний	Дійсне (подв. точність)	З кінця	За зростанням	Лінійний
13	Бульбашковий. Шелла	Довге ціле	З початку	За убуванням	Лінійний
14	Вставки. Швидкий	Символьне	З початку	За зростанням	Бінарний
15	Шелла. Швидкий	Дійсне	З кінця	За убуванням	Бінарний

Контрольні питання

1. Які групи методів сортування ви знаєте? В чому їхсутність?
2. Поясніть алгоритм методу сортування бульбашками.
3. Наведіть приклад приклад фрагмента коду, що здійснює сортування символів бульбашками у порядку зростання (починати сортування з першого елементу).
4. Наведіть схему роботи функції для реалізації бульбашкового сортування (сортування починати з кінця).
5. Поясніть алгоритм методу сортування вставками.
6. Наведіть схему роботи функції для реалізації сортування вставками (сортування починати з початку).
7. Наведіть приклад приклад фрагмента коду, що здійснює сортування цілих чисел вставками у порядку убування (починати сортування з першого елементу).
8. Поясніть алгоритм методу сортування вибором.
9. Наведіть схему роботи функції для реалізації сортування вибором (сортування починати з початку).
10. Наведіть приклад приклад фрагмента коду, що здійснює сортування дійсних чисел вибором у порядку убування (починати сортування з кінця).
11. Наведіть схему роботи функції для реалізації бульбашкового сортування.
12. Як оцінюються методи сортування?
13. Які удосконалення методів сортування ви знаєте?
14. В чому сутність шейкерного сортування? Наведіть приклад.
15. В чому сутність методу швидкого сортування? Його переваги.
16. Наведіть алгоритм методу Шелла для сортування? Його переваги.

ЛАБОРАТОРНА РОБОТА №11

ВИКОРИСТАННЯ ДИНАМІЧНИХ СТРУКТУР

Мета роботи

- Отримати практичні навички у роботі з динамічними структурами.
- Дослідити правила та можливості виконання операцій з динамічними структурами.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Якщо елементи зв'язані між собою за допомогою покажчиків, то такий спосіб організації даних називається ***динамічними структурами даних***, їх розмір динамічно змінюється під час виконання програми. З динамічних структур даних найчастіше використовуються лінійні списки, стеки, черги та бінарні дерева.

Лінійні списки

Лінійний список – це скінченна послідовність однотипних елементів (вузлів). Кількість елементів у цій послідовності називається довжиною списку. Наприклад, F=(1,2,3,4,5,6) – лінійний список, його довжина 6.

При роботі зі списками часто доводиться виконувати такі операції:

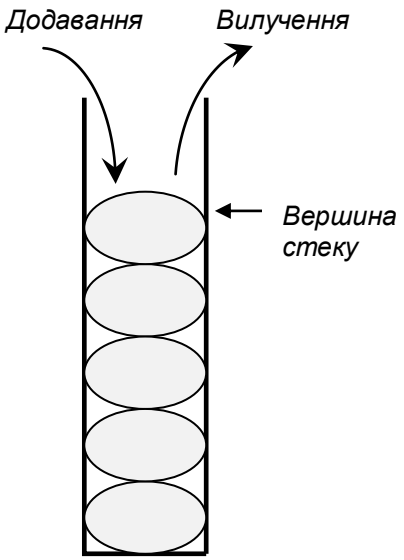
- додавання елемента в початок списку;
- вилучення елемента з початку списку;
- додавання елемента в будь-яке місце списку;
- вилучення елемента з будь-якого місця списку;
- перевірку, чи порожній список;
- очистку списку;
- друк списку.

Основні методи зберігання лінійних списків поділяються на методи послідовного та зв'язаного зберігання.

Послідовне зберігання списків. Метод послідовного зберігання списків ґрунтується на використанні масиву елементів деякого типу та змінної, в якій зберігається поточна кількість елементів списку. При послідовному зберіганні списків за допомогою масивів елементи списку зберігаються в масиві. Така організація дозволяє легко переглядати вміст списку та додавати нові елементи в його кінець. Але такі операції, як вставляння нового елементу в середину списку чи вилучення елементу з середини списку потребують зсуву всіх наступних елементів. При збільшенні елементів масиву кількість операцій для впорядкування списку стрімко зростає.

Зв'язане зберігання лінійних списків. Найпростіший спосіб зв'язати множину елементів – зробити так, щоб кожен елемент містив посилання на наступний. Це односпрямований (однозв'язаний) список. Якщо додати в такий список ще й посилання на попередній елемент, то отримаємо двозв'язаний список. А список, перший та останній елементи якого зв'язані, називається ***кільцевим***.

Стеки



Стек – динамічна структура даних, яка являє собою впорядкований набір елементів, в якому додавання нових елементів і видалення існуючих проходить з одного кінця – *вершини стеку*.

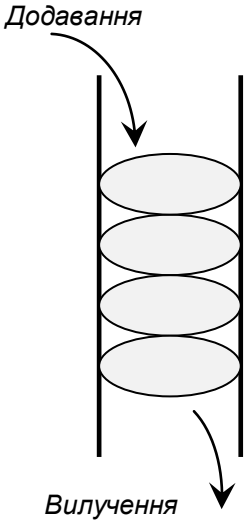
Стек реалізує принцип *LIFO* (last in – first out, останнім прийшов – першим пішов). Найбільш наглядним прикладом організації стеку може бути дитяча пірамідка, де додавання і знімання кілець здійснюється як раз відповідно до цього принципу.

Основні операції над стеками:

- додавання елемента в стек;
- вилучення елемента із стека;
- перевірка, чи порожній стек;
- перегляд елемента у вершині стека без видалення;
- очистка стека.

Стек створюється так само, як і лінійний список, оскільки стек є частковим випадком односпрямованого списку.

Черги



Черга – це лінійний список, де елементи вилучаються з початку списку, а додаються в кінець (як звичайна черга в магазині).

Двостороння черга – це лінійний список, у якого операції додавання, вилучення і доступу до елементів можливі як спочатку так і в кінці списку. Таку чергу можна уявити як послідовність книг, що стоять на полиці так, що доступ до них можливий з обох кінців.

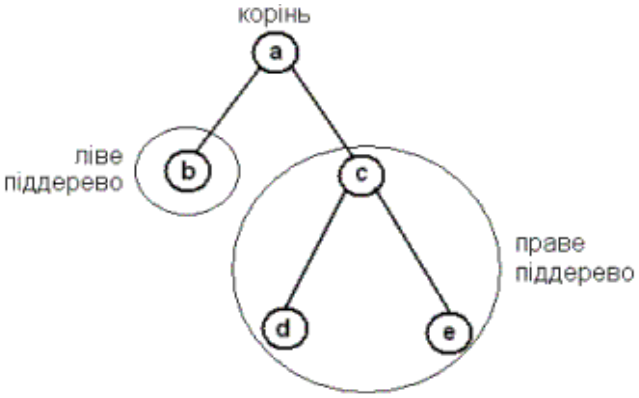
Черга є частковим випадком односпрямованого списку. Вона реалізує принцип *FIFO (*first in – first out, першим прийшов – першим пішов).

Черги створюються аналогічно до лінійних списків та стеків.

Двійкові дерева

Бінарне дерево – це динамічна структура даних, що складається з вузлів (елементів), кожен з яких містить, окрім даних, не більше двох

посилань на інші бінарні дерева. На кожен вузол припадає рівно одне посилання. Початковий вузол називається *коренем дерева*.



Якщо дерево організоване таким чином, що для кожного вузла всі ключі його лівого піддерева менші за ключ цього вузла, а всі ключі його правого піддерева – більші, воно називається *деревом пошуку*. Однакові ключі в деревах пошуку не допускаються.

В дереві пошуку можна знайти елемент за ключем, рухаючись від кореня і переходячи на ліве або праве піддерево в залежності від значення ключа в кожному вузлі. Такий спосіб набагато ефективніший пошуку по списку, так як час виконання операції пошуку визначається висотою дерева.

Дерево є рекурсивною структурою даних, так як кожне піддерево є також деревом. Дії з такими структурами даних простіше всього описувати за допомогою рекурсивних алгоритмів, хоча можна створити і його нерекурсивний еквівалент.

Для бінарних дерев визначені наступні операції:

- включення вузла у дерево;
- пошук по дереву;
- обхід дерева;
- видалення вузла.

Порядок виконання роботи

1. Засвоїти теоретичний матеріал. Використовуючи лекційний матеріал і інші літературні джерела, дослідити функції роботи з різними динамічними структурами.
2. Розробити програму згідно індивідуального завдання. Для кожного режиму роботи програми повинні бути розроблені окремі функції. Вхідні і вихідні дані повинні зберігатись у файлах.
3. Підготувати звіт:
   - варіант і текст завдання;
   - схема взаємодії функцій програми (з вказанням даних, що передаються і повертаються функціями);
   - схема даних програми;
   - лістинг програми і результати виконання;
   - висновки.

Варіанти індивідуальних завдань

Створити динамічну структуру, елементами якої є дані, що створювались і оброблялись у лабораторних роботах №7 та №9.

Реалізувати різні функції:

- додавання нових елементів у структуру;
- видалення елементу із заданим номером зі структури;
- виведення на екран інформації, яка зберігається у заданому елементі;
- виведення на екран усієї інформації зі структури.

№ варіанту	Вид динамічної структури
1, 5, 9, 13	Бінарне дерево
2, 6, 10, 14	Черга
3, 7, 11, 15	Стек
4, 8, 12, 16	Однозв’язаний список

Контрольні питання

1. Що таке динамічні структури? З чого вони складаються?
2. Наведіть різни приклади опису динамічних структур.
3. Які види динамічних структур існують?
4. Які види зберігання лінійних списків ви знаєте?
5. Які операції над лінійними списками можна виконувати?
6. Поясніть, що таке стеки.
7. Назвіть основні операції над стеками і поясніть їх виконання схематично.
8. Як можна отримати доступ до будь-якого елементу стека, окрім вершини?
9. Наведіть фрагмент коду для додавання елементу у стек.
10. Поясніть, що таке черги і правила їх організації.
11. Назвіть основні операції над чергами і поясніть їх виконання схематично.
12. Наведіть фрагмент коду для ініціалізації черги.
13. В чому сутність бінарного дерева?
14. Які операції над елементами бінарного дерева можна здійснювати?
15. Покажіть на прикладі створення бінарного дерева, елементи якого цілі числа або символи абетки.

ЛАБОРАТОРНА РОБОТА №12

КЛАСИ ЯК ОСНОВА ООП. ОБ'ЄКТИ ТИПУ КЛАС

Мета роботи

- Порівняти об'єктно-орієнтований та функціональний підходи у програмуванні.
- Ознайомитись на практиці з класами, об'єктами та головними елементами об'єктного підходу.
- Навчитись створювати і використовувати об’єкти типу клас.
- Навчитись на практиці застосовувати успадкування класів, створювати ієрархію класів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Класи та об'єкти в С++

Класи та об'єкти в С ++ є основними концепціями об'єктно-орієнтованого програмування (ООП). Об'єктно-орієнтоване програмування – розширення структурного програмування, в якому основними концепціями є поняття класів і об'єктів. Основна відмінність мови програмування С++ від С полягає в тому, що в С немає класів, а отже мова С не підтримує ООП, на відміну від С++.

Класи в С++ – це абстракція, що описує методи і властивості ще не існуючих об'єктів. ***Об'єкти*** – конкретне уявлення абстракції, що має свої властивості та методи. Створені об'єкти на основі одного класу називаються ***екземплярами*** цього класу. Ці об'єкти можуть мати різну поведінку, властивості, але все одно будуть об'єктами одного класу. В ООП існує три основних принципи побудови класів:

1. ***Інкапсуляція*** – це властивість, що дозволяє об'єднати в класі і дані, і методи, які працюють з ними і приховати деталі реалізації від користувача.
2. ***Успадкування*** – це властивість, що дозволяє створити новий клас-нащадок на основі вже існуючого, при цьому всі характеристики класу батька присвоюються класу-нащадку.
3. ***Поліморфізм*** – властивість класів, що дозволяє використовувати об'єкти класів з однаковим інтерфейсом без інформації про тип і внутрішній структурі об'єкта.

Оголошення класів в С++

```
class // ім'я класу
{
   private:
```

```
        // Список властивостей і методів для використання всередині
   класу
      public:
        // Список методів доступних іншим функціям та об'єктам програми
      protected:
        // список засобів, доступних при спадкуванні
    };
```

Приклад. Програма, в якій оголошено найпростіший клас, в якому оголошена одна функція, яка друкує повідомлення.

```
#include <iostream>
using namespace std;
     // оголошення класу
class CppClass
{
  public: // специфікатор доступу
     void message () // функція виводить повідомлення на екран
     {
         cout << "  Classes and Objects in C++ \n\n\n";
     }
};    // Кінець оголошення класу CppStudio

int main (int argc, char * argv [])
{
     CppClass obj;          // Оголошення об'єкта
     obj.message();         // Виклик функції класу message
     system("pause");
     return 0;
}
```

set-функції та get-функції класів

Кожен об'єкт має якісь свої властивості або атрибути, які характеризують його впродовж усього життя. Атрибути об'єкта зберігаються в змінних, оголошених всередині класу, якому належить даний об'єкт. Причому, оголошення змінних повинно виконуватися зі специфікатором доступу *private*. Такі змінні називаються елементами даних або полями класу. Оскільки елементи даних оголошені в *private*, то і доступ до них можуть отримати тільки методи класу, зовнішній доступ до елементів даних заборонений. Тому прийнято оголошувати в класах спеціальні методи – так звані *set* і *get* функції, за допомогою яких можна маніпулювати елементами даних. *set*-функції ініціалізують елементи даних, *get*-функції дозволяють переглянути значення елементів даних.

Доопрацюємо клас *CppClass* так, щоб у ньому можна було зберігати дату в форматі *дд.мм.гг*. Для зміни і перегляду дати реалізуємо *set*- і *get*-функції.

```
class CppClass // ім'я класу
{
   private:
     int day,   // день
         month, // місяць
         year;  // рік
   public:
```

```
      void message () // функція виводить повідомлення на екран
      {   cout << "\n\tClasses and Objects in C++ \n\n";
      }
      void setDate (int date_day, int date_month, int date_year)
      {   day = date_day;         // Ініціалізація дня
          month = date month;     // Ініціалізація місяця
          year = date_year;       // Ініціалізація року
      }
      void getDate () // відобразити поточну дату
      {
        cout<<"\n\tDate:"<<day<<"."<<month<<"."<<year<<endl<<endl;
      }
};          // Кінець оголошення класу CppStudio

int main ()
{     setlocale (LC_ALL, "rus"); // Установка локалі
      int day, month, year;
      cout << "\n\tВведіть поточний день, місяць і рік: \n\n";
      cout << "\tдень  : "; cin >> day;
      cout << "\tмісяць: "; cin >> month;
      cout << "\tрік   : "; cin >> year;
      CppClass obj;          // Оголошення об'єкта
      obj.message ();        // Виклик функції класу message
      obj.setDate (day, month, year); // Ініціалізація дати
      obj.getDate ();        // Відобразити дату
      system ("pause");
      return 0;
}
```

Результат:

```
Введіть поточний день, місяць і рік:

день  : 15
місяць: 12
рік   : 2014

Classes and Objects in C++


Date:15.12.2014
```

У визначенні класу специфікатор доступу *private* обмежує доступ до змінних, які оголошені після нього і до початку специфікатора доступу *public*. Таким чином, до змінних *day*, *month*, *year*, можуть отримати доступ тільки методи класу. Функції, що не належать класу, не можуть звертатися до цих змінних. Дані або методи класу, оголошені після специфікатора доступу *private*, але до початку наступного специфікатора доступу називаються *закритими елементами даних* і *закритими методами класу*.

Доцільно оголошувати елементи даних після специфікатора доступу *private*, а методи класу – після специфікатора *public*. Тоді, для маніпулювання елементами даних, оголошуються спеціальні функції – *get* і *set*.

В клас *CppClass* ми додали два методи *setDate()* і *getDate()*. Метод *setDate()* (*set*-функція) инициализирует елементи даних. Тобто метод *setDate()* ініціалізує змінні *day*, *month*, *year*.

Щоб переглянути, значення закритих елементів даних, оголошена функція *getDate()* (*get*-функція), яка повертає значення з змінних *day*, *month*, *year* у вигляді дати.

Конструктори і деструктори

Коли ми створюємо елементи (змінні) класу, ми не можемо присвоїти їм значення у самому визначенні класу. Компілятор видасть помилку. Тому нам необхідно створювати окремий метод (так звану *set*-функцію) класу, за допомогою якого і буде відбуватися ініціалізація елементів. При цьому, якщо необхідно створити, наприклад, 20 об'єктів класу, то прийдеться 20 разів викликати *set*-функції. Тут нам якраз зможе допомогти конструктор класу.

Конструктор (construct – створювати) – *це спеціальний метод класу, призначений для ініціалізації елементів класу деякими початковими значеннями.*

Деструктор (destruct – руйнувати) – *спеціальний метод класу, який служить для знищення елементів класу.* Найчастіше його використовують тоді, коли в конструкторі, при створенні об'єкта класу, динамічно була виділена ділянка пам'яті і необхідно цю пам'ять очистити, якщо ці значення вже не потрібні для подальшої роботи програми.

Важливо пам'ятати:

1) конструктор і деструктор завжди оголошуємо в розділі *public*;
2) при оголошенні конструктора тип повернення не вказується, в тому числі – void !!!;
3) у деструктора так само немає типу повернення, деструктору не можна передавати ніяких параметрів;
4) ім'я класу і конструктора повинні бути ідентичними;
5) ім'я деструктора ідентично імені конструктора, але з приставкою ~;
6) у класі допустимо створювати декілька конструкторів, якщо це необхідно. Імена будуть однаковими, а компілятор буде їх розрізняти по переданим параметрами (перевантаження функцій). Якщо ми не передаємо в конструктор параметри, він вважається конструктором за замовчуванням;
7) у класі може бути оголошений лише один деструктор.

Приклад.

```
class AB //класс
{
    private:
           int a;
           int b;
    public:
    AB()    //це конструктор
    {   a = 0; b = 0;      //початкові значення полів
        cout << "\n\n\tРобота конструктора: " << endl;
        cout << "\ta = " << a << endl;
        cout << "\tb = " << b << endl << endl;
    }
    void setAB() // змінюємо початкові значення
    {   cout << "\tВведите целое число a: ";  cin >> a;
        cout << "\tВведите целое число b: ";  cin >> b;
```

```
    }
    void getAB()
    {   cout << "\ta = " << a << endl;
        cout << "\tb = " << b << endl << endl;
    }
};

int main()
{   setlocale(LC_ALL, "rus");
    AB obj1;          //спрацює конструктор для I об’єкта
    obj1.setAB();     //задаємо нові значення
    obj1.getAB();     //виводимо їх на екран
    AB obj2;          //спрацює конструктор для II об’єкта
     system("pause");
     return 0;
}
```

Результат:

```
Работа конструктора при создании нового объекта:
a = 0
b = 0

Введите целое число a: 13
Введите целое число b: 17
a = 13
b = 17



Работа конструктора при создании нового объекта:
a = 0
b = 0
```

Успадкування класів

Упадкування класів дозволяє створювати похідні класи (класи спадкоємці), взявши за основу всі методи й елементи базового класу (класу батька). Таким чином, економиться маса часу на написання і налагодження коду нової програми. Об'єкти похідного класу вільно можуть використовувати все, що створено і налагоджено в базовому класі. При цьому ми можемо в похідний клас дописати необхідний код для удосконалення програми: додати нові поля, методи і т. д. Базовий клас залишиться недоторканим. Нижче наведено код програми, в якій створено два класи: базовий – *FirstClass* і похідний від нього *SecondClass*.

Приклад

```
class FirstClass     // базовий клас
{
protected:           // специфікатор доступу до элементу value
    int value;
public:
    FirstClass()       {  value = 0;    }
    FirstClass(int x)  {  value = x;    }
    void show_value()  {  cout << value << endl;  }
};

class SecondClass : public FirstClass   // похідний клас
{
public:
```

```
            // конструктор класу SecondClass викликає конструктори
класу
            // FirstClass
    SecondClass() : FirstClass() {}
    SecondClass(int inputS) : FirstClass(inputS) {}
    void ValueSqr() // Без специфікатора  protected не змогли б змінити value
    {
        value *= value;
    }
};

int main()
{   setlocale(LC_ALL, "rus");

    FirstClass F_object(3);     // об'єкт базового класу
    cout << "\n\tvalue F_object = ";
    F_object.show_value();

    SecondClass S_object(4);    // об'єкт похідного класу
    cout << "\tvalue S_object = ";
    S_object.show_value();      // виклик методу базового класу

    S_object.ValueSqr();        // підносимо value до квадрату
    cout << "\tквадрат value S_object = ";
    S_object.show_value();

    // F_object.ValueSqr();      // ПОМИЛКА: немає доступу
    cout << endl;
    system("pause");
    return 0;
}
```

Результат:

```
value F_object = 3
value S_object = 4
квадрат value S_object = 16
```

Основна інформація про успадкування класів:

1) Успадкування – це визначення похідного класу, який може звертатися до всіх елементів і методів базового класу, за винятком тих, що перебувають у розділі *private*.
2) Похідний клас ще називають нащадком або підкласом, а базовий – батьківським, або надкласом, або суперкласом.
3) Синтаксис визначення похідного класу:

class Імя_Похідн_Класу: специфікатор доступу Імя_Баз_Класу { . . . };

4) Похідний клас має доступ до всіх полів і методів базового класу, а базовий клас може використовувати тільки свої власні поля і методи.
5) У похідному класі необхідно явно визначати свої конструктори, деструктори і перевантажені оператори присвоювання через те, що вони не успадковуються від базового класу. Але їх можна викликати явним чином при визначенні конструктора, деструктора або перевантаження оператора присвоєння похідного класу, наприклад, таким чином (для конструктора):

Конструктор_Похідн_Класу (...): Конструктор_Баз_Класу (...) {...}.

Порядок виконання роботи

1. Засвоїти теоретичний матеріал. Використовуючи лекційний матеріал і інші літературні джерела, навчитись описувати класи, засвоїти способи створення об’єктів.
2. При розробці програми використати знання, отримані під час виконання лабораторних робіт: №7 (розробка меню), №9 (багатофайлові програми), №10 (сортування і пошук).
3. Підготувати звіт з лабораторної роботи, у якому представити такі матеріали:
   – зміст завдання і варіант;
   – лістинг програми.
   – схему ієрархії класів програмного результати виконання – вигляд екрану при виконанні програми;
   – результати роботи розробленого програмного засобу;
   – висновки.

Варіанти індивідуальних завдань

Завдання 1. Створити клас для обробки записів бази даних у відповідності з вказаною предметною областю. Розробити програму, дотримуючись таких вимог:

1) Розмістити інтерфейс класу у заголовочному файлі, а визначення функцій та головну функцію програми – у двох окремих файлах.
2) Передбачити можливість роботи з довільним числом записів, а також реалізувати окремими функціями класу:
   – конструктори без параметрів та з параметрами;
   – додавання об’єктів;
   – знищення об’єктів;
   – виведення інформації на екран;
   – пошук потрібної інформації за конкретною ознакою;
   – редагування записів;
   – сортування за різними полями.
3) При розробці програми слід здійснити захищення даних (опис з модифікатором *private*) для ізоляції елементів-даних класу від підпрограм, в яких цей клас використовується.
4) Програма повинна містити меню для перевірки всіх методів класу. Бажано для реалізації меню розробити окрему функцію, яка повертає номер вибраного пункту меню.

№	Предметна область	Реквізити об’єкту	Параметр сортування	Параметр пошуку
1.	Бібліотека	інвентарний номер, автор, назва, кількість сторінок, рік видання	Рік видання	Автор
2.	Телефонний довідник	Прізвище, ім'я, по батькові, домашня адреса, телефон.	Телефон	Прізвище
3.	Розклад руху літаків	Номер рейсу, тип літака, напрямок руху, періодичність вильоту.	Номер рейсу	Тип літака
4.	Колекція компакт-дисків	Інвентарний номер, назва альбому, об'єм диску, тип, дата запису.	Дата запису	Назва альбому
5.	Записна книжка	Прізвище, ім'я, домашня адреса, телефон, електронна пошта.	Прізвище	Електронна пошта
6.	Предметний покажчик	Слово; номера сторінок, де це слово зустрічається; кількість цих слів на даній сторінці	Номер сторінки	Слово
7.	Розклад пар	Номер пари, предмет, прізвище викладача, форма заняття.	Предмет	Номер пари
8.	Список файлів	ім'я файла, розширення, розмір, дата створення, атрибути.	Розширен-ня	Дата створення
9.	Архів програм	Назва програми, операційна система, розмір програми, дата запису	Назва програми	Операційна система
10.	Рахунки банку	Прізвище, ім'я, дата останньої операції, сума вкладу	Сума вкладу	Дата операції
11.	Користувачі локальної мережі	Прізвище, група, обліковий запис, тип облікового запису.	Тип облікового запису	Прізвище
12.	Камера схову	Прізвище, дата здачі, термін зберігання, інвентарний номер та назва предмета	Інвентар-ний номер	Дата здачі
13.	Склад товарів	інвентарний номер, назва товару, вага, ціна, кількість	Вага	Назва товару
14.	Каса продажу квитків	Назва пункту, час відправлення, дата відправлення, час прибуття, дата прибуття, ціна квитка	Час від-правлення	Назва пункту
15.	Успішність студентів	Прізвище, номер групи, оцінки з трьох предметів	Прізвище	Номер групи

Завдання 2. Утворити похідний клас, залучивши до нього як мінімум два додаткових поля таким чином, щоб клас набув більшої спеціалізованості. Для похідного класу використати конструктор, щоб він містив усі аргументи, необхідні для ініціалізації об'єкту похідного класу. Створити додаткові необхідні функції, що дозволяють перевірити роботу похідних класів.

Контрольні питання

1. Що таке об’єктно-орієнтоване програмування?
2. Яка різниця між об'єктно-орієнтованним та функціональним програмуванням?
3. Що таке класи? Який синтаксис їх опису?
4. Що таке об’єкти та екземпляри класу?
5. Які основні принципи в об’єктно-орієнтованому програмуванні? Назвіть їх і дайте коротку характеристику сутностей цих принципів.
6. Наведіть приклад опису будь-якого класу та приклад створення екземпляру цього класу.
7. Для чого потрібні *set*- і *get*-функції?
8. Які специфікатори доступу використовуються в описах класів? Поясніть їх значення.
9. Дайте поняття закритих полів і методів класу.
10. Що таке конструктори і деструктори класу? Для чого їх використовують?
11. Які особливості конструкторів у класах?
12. Які особливості деструкторів у класах?
13. Поясніть на прикладах сутність успадкування класів.
14. Наведіть синтаксис опису успадкованого класу. Поясніть на прикладах.
15. Що Ви можете сказати про модифікатори доступу в успадкованих класах?
16. Як успадковуються конструктори у похідних класах?
17. Які аиди успадкування Ви знєте? На ведіть приклади.
18. Що таке віртуальні функції? Наведіть приклади використання.
19. Що таке дружні функції і дружні класи?

ЛАБОРАТОРНА РОБОТА №13

ПОЛІМОРФІЗМ. ПЕРЕВАНТАЖЕННЯ ФУНКЦІЙ, ОПЕРАТОРІВ І МЕТОДІВ КЛАСУ

Мета роботи

- Засвоїти правила створення перевантажених функцій.
- Навчитись перевантажувати унарні та бінарні оператори.
- Навчитись на практиці перевантажувати функції і операції у класах.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Перевантаження функцій

При визначенні функцій в своїх програмах необхідно вказати тип значення, що повертається функцією, а також кількість параметрів і тип кожного з них. Припустимо, є функція з ім'ям *add_values*, яка працює з двома цілими значеннями, а нам треба використовувати подібну функцію для додавання трьох цілих значень. Тоді слід створити функцію з іншим ім'ям, наприклад, *add_two_values* і *add_three_values*. Аналогічно якщо ви хотіли використовувати подібну функцію для складання значень типу *float,* то була б необхідна ще одна функція з ще одним ім'ям.

Для уникнення дублювання функції C++ дозволяє вам визначати декілька функцій з одним і тим же ім'ям. У процесі компіляції C ++ бере до уваги кількість аргументів, що використовуються кожною функцією, і потім викликає саме потрібну опцію. *Надання компілятору вибору серед декількох функцій називається* ***перевантаженням функцій***.

Обмеження на перевантажені функції:

1. Будь-які перевантажені функції повинні мати різні списки параметрів (при цьому аргумент даного типу і посилання на цей тип розглядаються як одне те саме).
2. Не допускається перевантаження функцій із співпадаючими списками параметрів лише на основі типу повертаємих значень.
3. Функції-члени не можуть бути перевантажені лише на основі того, що одна з них – статична, а друга – ні.
4. Typedef- визначення не впливають на механізм перевантаження, оскільки вони не вводять нові типи, а є лише синонімами існуючих. Наприклад, якщо є визначення

```
typedef  char *ptr;
```

то дві функції

```
void setval (char *s)  та  void setval ( ptr s)
```

не є різними. Це буде помилкою.

5. Всі enum-типи розглядаються як різні і тому можуть використовуватись для перевантаження функцій.
6. Типи “масив (чогось)” і “покажчик (на щось)” розглядаються як ідентичні з точки зору перевантаження. Наприклад,

```
void setval(char pz);
void setval (char *ptr);
```

є помилкою при перевантаженні.

Але це стосується тільки одномірних масивів. Для багатомірних масивів друга, третя,..... розмірності розглядаються як частина типу даних.

Наприклад, не буде помилки:

```
void setval (char sz [ ]);
void setval (char sz [ ][4]);
```

Приклад 1. Наступна програма перевантажує функцію з ім'ям *add_values*. Перше визначення функції складає два значення типу int. Друге визначення функції складає три значення.

```
#include <iostream.h>

int add_values(int a,int b){
   return(a + b);
}
int add_values (int a, int b, int c){
   return(a + b + c);
}
void main(void){
   cout<<"200 + 801 = "<<add_values(200,801)<<endl;
   cout<<"10 + 21 + 70 = "<<add_values(10,21,70)<<endl;
}
```

Приклад 2. Наступна програма перевантажує функцію *show_message*. Перша функція виводить стандартне повідомлення, параметри їй не передаються. Друга виводить передане їй повідомлення, а третя виводить два повідомлення.

```
#include <iostream.h>

void show_message(void){  cout << "Стандартное сообщение: "
         << "Учимся программировать на C++" << endl;
}

void show_message(char *message){
   cout << message << endl;
}

void show_message(char *first, char *second){
   cout << first << endl;
   cout << second << endl;
}

void main(void){
    show_message();
   show_message("Учимся программировать на языке C++!");
   show_message("В C++ нет предрассудков!","Перегрузка-это круто!") ;
}
```

Перевантаження операторів

Перевантаження оператора полягає у зміні сенсу оператора при використанні його з певним класом. Перевантаження операторів може спростити найбільш загальні операції класу і поліпшити читабельність програми

Для перевантаження операторів програми використовують ключове слово ***operator.***

Якщо програма перевантажує оператор для певного класу, то сенс цього оператора змінюється тільки для зазначеного класу, решта програми буде використовувати цей оператор для виконання його стандартних операцій.

У загальному випадку програми можуть перевантажити майже всі оператори С ++, окрім таких:

- оператор вибору члена (оператор “.”);
- оператор вибору члена за покажчиком (оператор “.*”);
- оператор розширення області бачення (оператор “::”);
- оператор умови (“? : “);
- препроцесорний оператор (“#”) та препроцесорний символ (“##”).

Приклад 3. Нижче наведено визначення класу *strings*. Цей клас містить один елемент даних, який являє собою власне символьний рядок. Крім того, цей клас містить декілька перевизначених операторів:

```
#include <iostream>
#include <string.h>
#include <locale.h>
using namespace std;
class strings
{
  public:
    strings(char *); // Конструктор
    void operator +(char *);
    void operator -(char);
    void show_string(void);
  private:
    char data[256] ;
};
strings::strings(char *str){
   strcpy(data, str);
}
void strings::operator +(char *str){
   strcat(data, str);
}
void strings::operator -(char letter){
   char temp[256] ;
   int i, j;
   for (i = 0, j = 0; data[i]; i++)
         if (data[i]==letter) temp[j++] = data[i];
   temp[j] = NULL;
```

```
   strcpy(data, temp);
}
void strings::show_string(void){
   cout << "\n\t"<<data << endl;
}
void main(void){
   setlocale(0,"");
   strings title( "\n\tУчимся программировать на C++");
   strings lesson("\n\tПерегрузка операторов");
   title.show_string();
   title + " и я учусь тоже!";
   title.show_string() ;
   lesson.show_string();
   lesson – 'p';
   lesson.show_string();
   system("pause");
}
```

Порядок виконання роботи

1. Засвоїти теоретичний матеріал. Використовуючи лекційний матеріал і інші інформаційні джерела, навчитись створювати і перевантажувати функції, перевантажувати оператори для операцій над об’єктами класу.
2. Розібратись у наведених прикладах і здійснити компіляцію програм з прикладів 1,2,3.
3. Розробити програми згідно індивідуального завдання.
4. Підготувати звіт з лабораторної роботи, у якому представити такі матеріали (спільний звіт для усіх задач):
   - зміст завдання і варіант;
   - математична формалізація задачі (для завдання 2);
   - лістинг програми;
   - результати роботи розробленого програмного засобу в різних режимах;
   - висновки.

Варіанти індивідуальних завдань

Завдання 1. Визначившись з власними силами, вибрати собі для виконання одну із задач. При цьому номер задачі обрати таким чином:

- для задач низької складності: номер_задачі = номер_варіанту % 5;
- для задач середньої і високої складності:

  номер_задачі = номер_варіанту % 3.

Протестуйте визначені функції на тестових наборах даних.

(низька складність – оцінка "задовільно")	
1	Визначте перевантажені функції *square* для знаходження квадратів цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
2	Визначте перевантажені функції *triple* для потроювання значень цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
3	Визначте перевантажені функції *minimum* для знаходження найменшого з трьох цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
4	Визначте перевантажені функції *maximum* для знаходження найбільшого з трьох цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
(середня складність – оцінка "добре")	
1	Визначте перевантажені функції *power* для піднесення до цілого невід'ємного степеня цілих чисел, чисел з плаваючою комою одинарної та подвійної точності.
2	Визначте перевантажені функції *round_value* для округлення чисел з плаваючою комою подвійної точності до заданої точності. Точність може бути задана у вигляді цілого числа, що вимагає необхідне число знаків після коми, та у вигляді числа з плаваючою комою у форматі 0,0…01 (наприклад, 0,01 означає округлення до сотих).
(висока складність – оцінка "відмінно")	
1	Визначте перевантажені функції *power* для піднесення до цілого степеня цілих чисел, чисел з плаваючою комою одинарної та подвійної точності. Значення показника степеня може бути від'ємним, нульовим та додатнім.
2	Визначте перевантажені функції *round_value* для округлення чисел з плаваючою комою подвійної точності до заданої точності. Точність може бути задана у вигляді цілого числа, що вимагає необхідне число знаків після коми, та у вигляді числа з плаваючою комою у форматі 0,0…01 (наприклад, 0,01 означає округлення до сотих). Передбачте можливість округлення за необхідністю до першої значущої цифри у випадку, якщо округлене до заданої точності число рівне нулю.

Завдання 2. Описати клас, що реалізує вказаний тип даних згідно індивідуального завдання. Клас повинен містити:

- множину конструкторів для створення об'єктів певного типу (конструктор за замочуванням, конструктор з параметрами, конструктор копіювання);
- вказані операції над об'єктами класу з використанням механізму перевантаження операцій.

Написати програму, яка демонструє роботу з об'єктами створеного класу. Програма повинна містити меню для перевірки усіх методів класу і операцій. Організувати виведення та введення даних за допомогою класів-потоків *cin* та *cout*.

№	Предметна область	Операції для перевантаження
1	Матриця	Віднімання, множення на матрицю, додавання цілого числа
2	Комплексні числа	Сума, добуток на комплексне число, добуток на дійсне число інкремент
3	Вектор у просторі	Додавання векторів, векторний добуток двох векторів
4	Множина	Вилучення елемента, об'єднання множин, перетин множин
5	Точка на площині	Додавання двох точок, множення на число, дзеркальне відображення
6	Дроби	Віднімання, множення, порівняння
7	Лінія	Порівняння, зменшення, пересування по горизонталі
8	Цілі числа	Інкремент, декремент, додавання, віднімання, логічні операці
9	Вектор на площині	Скалярний добуток, порівняння векторів, множення вектора на число
10	Матриця	Додавання, множення на число, транспонування
11	Прямокутник	Збільшення, пересування по вертикалі, перевірка на попадання точки в прямокутник
12	Дроби	Додавання, інкремент, пошук оберненого дробу
13	Дійсне число	Модуль числа, сума, добуток на ціле, добуток на дійсне
14	Комплексні числа	Різниця, порівняння, ділення на дійсне число, декремент
15	Комплексні числа	Обчислення оберненого комплексного числа, додавання до комплексного числа дійсного і навпаки

Контрольні питання

1. Для чого використовують перевантаження функцій?
2. Які обмеження на перевантажені функції?
3. Як компілятор діє з перевантаженими функціями?
4. Наведіть приклад перевантаження функцій.
5. В чому сенс перевантаження операторів?
6. Наведіть приклади перевантаження унарних та бінарних операторів.
7. Які оператори не можна перевантажувати?
8. Для чого перевантажують конструктори класів?
9. Наведіть приклад перевантаження конструкторів.
10. Що таке конструктор копіювання? Наведіть приклад такого методу.
11. Чи можна перевантажувати деструктори?

ЛАБОРАТОРНА РОБОТА №14

ШАБЛОНИ ФУНКЦІЙ І КЛАСІВ. ПАРАМЕТРИЗОВАНІ КОНТЕЙНЕРНІ КЛАСИ БІБЛІОТЕКИ STL

Мета роботи

- Ознайомитися із базовими механізмами використання шаблонів функцій.
- Навчитись створювати та використовувати шаблонів класів.
- Засвоїти правила створення та використання параметризованих контейнерних класів.
- Дослідити основні класи з бібліотеки STL.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Шаблони функцій

Шаблони функцій – це потужний інструмент у C++, який суттєво спрощує роботу програміста. Наприклад, нам потрібно запрограмувати функцію, яка виводила б на екран елементи масиву. При цьому ми хочемо, щоб функція виводила масиви типу *int*, *double*, *float* і *char*. Тобто, нам потрібно запрограмувати 4 функції, які виконують одні й ті самі дії, але для різних типів даних. Скористаємося перевантаженням функцій.

```
void printArray(const int * array, int count)
{   for (int ix = 0; ix < count; ix++)
        cout << array[ix] << "   ";
    cout << endl;
}
void printArray(const double * array, int count)
{   for (int ix = 0; ix < count; ix++)
        cout << array[ix] << "   ";
    cout << endl;
}
void printArray(const float * array, int count)
{   for (int ix = 0; ix < count; ix++)
        cout << array[ix] << "   ";
    cout << endl;
}
void printArray(const char * array, int count)
{   for (int ix = 0; ix < count; ix++)
        cout << array[ix] << "   ";
    cout << endl;
}
```

Таким чином, ми маємо 4 перевантажені функції, для різних типів даних. Ввони відрізняються тільки заголовком, тіло у них абсолютно однакове.

А що якщо, нам знадобиться запрограмувати алгоритм сортування у вигляді функції. Виходить, для кожного типу даних доведеться свою функцію створювати. Тобто, один і той же код буде в декількох примірниках, що дуже нераціонально. Тому в С ++ придуманий такий механізм – шаблони функцій.

Шаблони функцій *– це інструкції, згідно з якими створюються локальні версії функції для певного набору параметрів і типів даних.*

Синтаксис:

```
template <class T>
template <typename T>
template <typename T1, typename T2>
```

Всі шаблони функцій починаються зі слова *template*, після якого йдуть кутові дужки, в яких перераховується список параметрів. Кожному параметру має передувати зарезервоване слово *class* або *typename*.

Ключове слово *typename* говорить про те, що у шаблоні буде використовуватися вбудований тип даних, такий як: *int*, double, float, *char* і т. д. А ключове слово *class* повідомляє компілятору, що в шаблоні функції як параметр будуть використовуватися типи даних користувача, тобто класи.

Ми створюємо один шаблон, в якому описуємо всі типи даних. Таким чином код не буде захаращуватися.

Приклад 1. Програма виводить на екран елементи масивів різних типів, використовуючи шаблон функції.

```
#include <iostream>
using namespace std;
    // шаблон функції printArray
template <typename T>
void printArray(const T * array, int count)
{   for (int ix = 0; ix < count; ix++)
        cout << array[ix] << "   ";
    cout << endl;
}
int main()
{    const int iSize = 10, dSize = 7, fSize = 10, cSize = 5;
          // масиви різних типів даних
    int    iArray[iSize]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    double dArray[dSize]={1.2345,2.234,3.57,4.67876,5.346,6.1545,7.7682};
    float
fArray[fSize]={1.34,2.37,3.23,4.8,5.879,6.345,73.434,8.82,9.33,10.4};
    char   cArray[cSize] = {"MARS"};
    cout << "\n\t Використання шаблонів функцій:\n";
    cout << "\n\tМасив типу int:\n\t";  printArray(iArray, iSize);
    cout << "\n\tМасив типу double:\n\t"; printArray(dArray, dSize);
    cout << "\n\tМасив типу float:\n\t";printArray(fArray, fSize);
    cout << "\n\tМасив типу char:\n\t"; printArray(cArray, cSize);
   system("pause");
    return 0;
}
```

Результат роботи програми:

```
 Використання шаблонів функцій:

Масив типу int:
1   2   3   4   5   6   7   8   9   10

Масив типу double:
1.2345   2.234   3.57   4.67876   5.346   6.1545   7.7682

Масив типу float:
1.34   2.37   3.23   4.8   5.879   6.345   73.434   8.82   9.33   10.4


Масив типу char:
M   A   R   S
```

Шаблони класів

У міру того, як кількість створюваних класів зростає, виявляється, що деякий клас, створений для однієї програми, необхідний в іншій програмі. У багатьох випадках класи можуть відрізнятися тільки типами. Наприклад, один клас працює з цілочисельними значеннями, в той час як в іншій програмі він повинен працювати зі значеннями типу float.

Щоб збільшити ймовірність повторного використання існуючого коду, C++ дозволяє програмам визначати шаблони класів.

Шаблон класу *визначає типонезалежний клас, який надалі служить для створення об'єктів необхідних типів.*

Якщо компілятор C++ зустрічає оголошення об'єкта, засноване на шаблоні класу, то для побудови класу необхідного типу він буде використовувати типи, зазначені при оголошенні. Дозволяючи швидко створювати класи, що відрізняються тільки типом, шаблони класів скорочують обсяг програмування, що, в свою чергу, заощаджує час.

Пояснимо використання шаблонів класів на простому прикладі. Нехай необхідно створити простий клас "Масив", який буде виконувати прості дії: Додавання і Відображення елементів.

Приклад. Створення класу "Масив" без шаблону.

```
#include <iostream>
using namespace std;
          /*НАШ КЛАСС*/
 class Massiv
 {
    int Array[10];     //Масив цілочислених значень з 10 елементів
    int count;         //Лічильник елементів масиву
  public:
    void Add(int );    //Метод для додавання елементів в масив
    void Show();       //Метод для відображення масиву на екрані
 };
 void Massiv::Add(int x)
 {
   static int pos=0;
   Array[pos]=x;
   pos++;
   count=pos;
```

```
 }
 void Massiv::Show()
 {
  for (int i=0;i<count;i++)
      cout<<Array[i]<<"\t";
  cout<<endl;
 }
int main()
{
  Massiv Arr;
  Arr.Add(100.555);
  Arr.Add(200);
  Arr.Add(300);
  Arr.Show();
  system("pause");
  return 0;
}
```

Це приклад створення звичайного класу. Але у програмістів іноді виникає необхідність створення такого ж класу, в якому відрізняється тільки тип даних. Наприклад, може знадобитися створення класу, в якому потрібно створення масиву, який буде зберігати в собі і обробляти не цілочисельні змінні, а рядкові. Як варіант, можна дописати купу класів для кожного з типів змінних, але це не раціонально. Код вийде більшим, громіздким. Чим більше коду, тим простіше в ньому помилятися і тим складніше шукати. Ось тут і приходять на допомогу шаблони класів.

Приклад 2. Створення класу "Масив" з використанням шаблону.

```
#include <iostream>
using namespace std;
int pos=0; //Позиція в масиві
template <class T> //Шаблон с класу з параметром T
 class Massiv
 {
    T Array[10];      //Масив цілочислених значень з 10 елементів
    int count;        //Лічильник елементів масиву
   public:
    void Add(T ); //Метод для додавання елементів в масив
    void Show();       //Метод для відображення масиву на екрані
 };
 template <class T> void Massiv<T>::Add(T x)
 { static int pos=0;
   Array[pos]=x;
   pos++;
   count=pos;
 }
 template <class T> void Massiv<T>::Show()
 {  cout<<"\t"<<endl;
    for (int i=0;i<count;i++)
          cout<<"\t"<<Array[i];
    cout<<endl;
 }
int main()
{
  setlocale(0,"");
```

```
    Massiv<int> Arr;
    Arr.Add(100.555);
    Arr.Add(200);
    Arr.Add(300);
    Arr.Show();
    Massiv<char *> Arr2;
    Arr2.Add("Строка");
    Arr2.Add("Начинаю понимать");
    Arr2.Add("УРА");
    Arr2.Show();
    cout<<endl;
    system("pause");
    return 0;
}
```

Результат роботи програми:

```
100     200     300

Строка  Начинаю понимать       УРА
```

Стандартна бібліотека шаблонів STL

Стандартна бібліотека шаблонів надає набір добре сконструйованих узагальнених компонентів C++. Бібліотека містить п'ять основних видів компонентів:

1) алгоритм (*algorithm*) – визначає обчислювальну процедуру;
2) контейнер (*container*) – управляє набором об'єктів в пам'яті;
3) ітератор (*iterator*) – забезпечує для алгоритмів засіб доступу до вмісту контейнера;
4) функціональний об'єкт (*function object*) – інкапсулює функцію в об'єкті для використання іншими компонентами;
5) адаптер (*adaptor*) – адаптує компонент для забезпечення різного інтерфейсу.

До найпопулярніших контейнерів відносять такі:

vector – колекція елементів, збережених в масиві, що збільшується в міру необхідності. Заголовочний файл – <*vector*>.

list – колекція елементів, збережених, як двонаправлений зв'язаний список. Заголовочний файл – <*list*>.

map – це колекція, яка зберігає пари значень pair <*Key,T*> і призначена для швидкого пошуку значення за ключем Key. Як ключ може бути використано, наприклад, рядок або *int*, але при цьому необхідно пам'ятати, що головною особливістю ключа є можливість застосувати до нього операцію порівняння. Важливо: ключ повинен бути унікальним. Заголовочний файл – <*map*>.

set – це колекція унікальних значень *Key* – кожне з яких є також і ключем. Тобто, це відсортована колекція, призначена для швидкого пошуку необхідного значення. До ключа пред'являються ті ж вимоги, що й у випадку ключа для *map*. Природно, використовувати її для цієї мети немає сенсу, якщо зберігати в ній прості типи даних, щонайменше

необхідно визначити свій клас, який зберігає пару ключ – значення і визначає операцію порівняння по ключу. Дуже зручно використовувати дану колекцію, якщо треба уникнути повторного збереження одного і того ж значення. Заголовочний файл – <*set*>.

multimap – це модифікований *map*, в якому відсутня вимоги унікальності ключа Тобто, якщо здійснювати пошук по ключу, то повернеться не одне значення, а набір значень, збережених з даними ключем. Заголовочний файл – <*map*>.

multiset – те ж саме відноситься і до цієї колекції, вимоги унікальності ключа в ній не існує, що призводить до можливості зберігання дублікатів значень. Тим не менш, існує можливість швидкого знаходження значень по ключу у випадку, якщо визначити свій клас. Оскільки всі значення в *map* і *set* зберігаються у відсортованому вигляді, то в цих колекціях можна швидко відшукати необхідне значення за ключем, але при цьому операція вставляння нового елемента буде складнішою, ніж, наприклад, в *vector*. Заголовочний файл – <*set*>.

Приклад 3. Використання контейнера *vector* з *STL* для цілих чисел.

```
#include <vector>
#include <algorithm>
#include <iostream>
using namespace std;
template <class T> void pr(T& v)
{         //===== Шаблон функції для виведення за допомогою ітератора

    T::iterator p;     //====== Ітератор для будь-якого контейнера
    for ( p=v.begin(),int i=0; p!=v.end(); p++, i++)
          cout << endl << i + 1 <<". "<< *p;
    cout << '\n';
}
void main ()
{   vector <int> v(10);      //======== Вектор цілих
    cout << "\nInt Vector:\n";
    for (int i=0; i<v.size(); i++)
    {     v[i] = rand()%10 + 1;
          cout << v[i]<< "; ";
    }
    sort (v.begin (), v.end());    //======= Сортування за замовчуванням
    cout << "\n\nAfter default sort\n";
    for (int i=0; i<v.size(); i++) cout << v[i]<< "; ";
    cout << "\n\nUsing iterators\n\n";
    pr(v);
    v.erase(v.begin());      //======== Видалення елементів
    cout << "\n\nAfter first element erasure\n";
    for (int i=0; i<v.size(); i++) cout << v[i]<< "; ";
    v.erase (v.end()-2, v.end());
    cout << "\n\nAfter last 2 elements erasure\n";
    for (int i=0; i<v.size(); i++) cout << v[i]<< "; ";
    int size = 2;            //======== Зміна розмірів
    v.resize(size);
    cout << "\n\nAfter resize, the new size: " << v.size()<< endl;
    for (uint i=0; i<v.size(); i++) cout << v[i]<< "; ";
```

```
    v.resize(6,-1);
    cout << "\n\nAfter resize, the new size: " << v.size()<< endl;
    for (uint i=0; i<v.size(); i++) cout << v[i]<< "; ";
    cout << "\n\nVector's maximum size: " << v.max_size()
         << "\nVector's capacity: " << v.capacity() << endl;
    v.reserve (100);
    cout << "\nAfter reserving storage for 100 elements:\n"<< "Size: "
      <<v.size()<<endl<<"Maximum size: "<<v.max_size()<< endl
      << "Capacity: " << v.capacity() << endl;
    v.resize(2000);
    cout << "\nAfter resizing storage to 2000 elements:\n"<<"Size:
"<<v.size()
      <<endl<<"Maximum size: "<<v.max_size() << endl<<"Capacity: "
      <<v.capacity()<<endl<< "\n\n";
    _getch();
}
```

Результат роботи програми:

```
Int Vector:
2; 8; 5; 1; 10; 5; 9; 9; 3; 5;

After default sort
1; 2; 3; 5; 5; 5; 8; 9; 9; 10;

Using iterators


1. 1
2. 2
3. 3
4. 5
5. 5
6. 5
7. 8
8. 9
9. 9
10. 10
```

```
After first element erasure
2; 3; 5; 5; 5; 8; 9; 9; 10;

After last 2 elements erasure
2; 3; 5; 5; 5; 8; 9;

After resize, the new size: 2
2; 3;

After resize, the new size: 6
2; 3; -1; -1; -1; -1;

Vector's maximum size: 1073741823
Vector's capacity: 10

After reserving storage for 100 elements:
Size: 6
Maximum size: 1073741823
Capacity: 100

After resizing storage to 2000 elements:
Size: 2000
Maximum size: 1073741823
Capacity: 2000
```

Порядок виконання роботи

4. Засвоїти теоретичний матеріал. Використовуючи лекційний матеріал і інші інформаційні джерела, навчитись створювати шаблони функцій і класів.
5. Розібратись у наведених прикладах і здійснити компіляцію програм з прикладів 1,2,3.
6. Розробити програми згідно індивідуального завдання.
5. Підготувати звіт з лабораторної роботи, у якому представити такі матеріали:
   - зміст завдання і варіант;
   - лістинг програми;
   - результати роботи розробленої програми;
   - висновки.

Варіанти індивідуальних завдань

Створити клас, який описує та забезпечує дії над даними параметризованого масиву, розмірність якого визначається під час роботи програми. Усі обчислення і перетворення реалізувати у вигляді функцій-членів класу.

1	В масиві обчислити: – номер елемента масиву, найближчого до середньоарифметичного його значень; – різниця елементів масиву, що розташовані між першим від'ємним та другим додатним елементами. Перетворити масив так, щоб у його першій половині розташовувались елементи, що стоять в парних позиціях, а в другій – елементи, що стоять в непарних позиціях.
2	Дана прямокутна матриця. Визначити: – кількість від'ємних елементів в тих рядках, які містять хоча б один нульовий елемент; – суму модулів елементів, які розташовані після першого додатного елемента Впорядкувати елементи матриці за спаданням модулів елементів
3	У довільній матриці обчислити: – кількість елементів масиву, рівних нулю; – суму елементів масиву, які лежать в діапазоні від A до B. Впорядкувати елементи масиву за спаданням модулів елементів
4	У довільній матриці обчислити: – кількість елементів масиву, рівних нулю; – суму елементів масиву, які лежать в діапазоні від A до B. Впорядкувати елементи масиву за спаданням модулів елементів
5	В одновимірному масиві елементів, обчислити: – номер максимального за модулем елемента; – суму модулів елементів, які розташовані після першого додатного елемента. Перетворити масив таким чином, щоб спочатку розташовувались всі елементи, ціла частина яких лежить в інтервалі [a,b], а потім – всі інші
6	В масиві обчислити: – мінімальний за модулем елемент масиву; – суму модулів елементів, які розташовані після першого від'ємного елемента. Ущільнити масив, видаливши з нього елементи, величина яких знаходиться на інтервалі [a,b]. Місце, яке звільниться в кінці масиву заповнити символом чи числом з клавіатури.
7	В масиві обчислити: – мінімальний за модулем елемент масиву; – суму модулів елементів масиву, розташованих після першого нільового елемента. Перетворити масив так, щоб в першій його половині розташовувались елементи, що стоять на парних позиціях, а в другій – елементи, що стоять в непарних позиціях.
8	У матриці обчислити: – максимальний за модулем елемент масиву; – суму елементів масиву, що розташовані між першим і другим додатними елементами. Перетворити матрицю так, щоб всі елементи, рівні нулю, розташовувались в кінці.

9	Дана прямокутна матриця. Визначити: – кількість рядків, які не містять жодного нульового елемента; – максимальне із чисел, що зустрічається в заданій матриці більше одного разу Перетворити матрицю так, щоб всі нульові елементи розташовувались на початку.
10	Дана прямокутна матриця. Визначити: – кількість стовпців, які не містять жодного нульового елемента. – кількість елементів, менших за A , але більших за B. Переставляючи рядки заданої матриці, розташувати їх у відповідності із зростанням суми значень у стовпцях.
11	В одномірному масиві обчислити: – добуток елементів масиву з парними номерами; – суму елементів масиву, які розташовані між першим і останнім нульовими елементами. Впорядкувати масив таким чином, щоб спочатку розташовувались всі додатні елементи, а потім – всі від'ємні (елементи, рівні 0 вважати додатними).
12	Дана прямокутна матриця. Визначити : – кількість стовпців, які містять хоча б один нульовий елемент; – номер рядка, в якому знаходиться найдовша серія з однакових елементів. Впорядкувати масив таким чином, щоб спочатку розташовувались всі серії з однакових елементів, а потім – всі решта елементів.
13	В одномірному масиві, що складається з N дійсних елементів, обчислити: – суму елементів масиву з непарними елементами; – суму елементів масиву, які розташовані між першим і останнім від'ємними елементами. Перетворити масив, видаливши з нього всі елементи, модуль яких не перевищує число, що вводиться з клавіатури. Елементи, які звільняться в кінці масиву заповнити нулями.
14	Дана прямокутна матриця. Визначити : – добуток елементів в тих рядках, які не містять від'ємних елементів; – максимум серед сум елементів діагоналей, паралельних головній діагоналі матриці. Перетворити матрицю, видаливши з неї всі елементи, модуль яких не перевищує число, що вводиться з клавіатури. Елементи, які звільняться в кінці масиву, заповнити нулями.
15	В одномірному масиві, що складається з N дійсних елементів, обчислити: – максимальний елемент масиву; – суму елементів масиву, що розташовані до останнього додатного елемента. Перетворити масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі [a,b]. Елементи, які звільняються в кінці масиву заповнити нулями.

Контрольні питання

1. Яка різниця між шаблоном та макросом?
2. Чим відрізняється параметр шаблона від параметру функції?
3. Як створити шаблонний клас?
4. Що являє собою стандартна бібліотека шаблонів STL?
5. Які основні шаблонні класи колекцій ви знаєте?
6. Що представляють собою ітератори?

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Пахомов Б. C/C++ и MS Visual C++ 2010 для начинающих. – СПб.: БХВ-Петербург, 2011. – 736 с. ISBN: 978-5-9775-0599-4.
2. Глушаков С. В. Язык программирования C++ / Глушаков С. В., Коваль А. В., Смирнов С. В. – Харьков: Фолио, 2001. – 500 с. – ISBN 966-03-1282-2.
3. Дейтел Х. М. Как программировать на C: 3-е издание / Х. М. Дейтел, П. Дж. Дейтел ; пер. с англ. В. В. Тимофеев. – М: Бином-Пресс, 2002. – 1168 с. – ISBN 5-9518-0002-1 (в пер.).
4. Дейтел Х. М. Как программировать на C++: 5-е издание / Х. М. Дейтел, П. Дж. Дейтел ; пер. с англ. В. В. Тимофеев. – М: Бином-Пресс, 2008. – 1456 с. – ISBN 978-5-9518-0224-8 (в пер.).
5. Єжова Л. Ф. Алгоритмізація і програмування процедур обробки інформації: Навч.- метод. посібник для самост. вивч. дисц. / Л. Ф. Єжова. – К.: КНЕУ, 2000. – 152 с. – ISBN 966–574–133–0.
6. Кнут Д. Е. Исскуство программирования, том 1. Основные алгоритмы, 3-е изд. / Д. Е. Кнут. – М.: «Вільямс», 2001. – 720 с. – ISBN 5-8459-0080-8 (в пер.).
7. Кнут Д. Е. Исскуство программирования, том 3. Сортировка и поиск, 2-е изд. / Д. Е. Кнут. – М.: «Вільямс», 2001. – 832 с. – ISBN 5-8459-0082-4 (в пер.).
8. Искусство программирования на C. Фундаментальные алюритмы, структуры данных и примеры приложений. Энциклопедия программиста / [Хэзфилд Р., Кирби Л. и др.] – М: Диасофт, 2001. – 736 с. – ISBN 966-7393-82-8 (в пер.).
9. Шилдт Г., Полный справочник по C / Г. Шилдт. – СП.: Вильямс, 2003. – 800 с. – ISBN 5-8459-0226-6 (в пер.).
10. Алгоритмы: построение и анализ, 2-е издание / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн ; пер. с англ. И. В. Красикова и др. – М. : Издательский дом "Вильямс", 2005. – 1296 с. – ISBN 5-8459-0857-4 (в пер.).
11. Лаптев В. В. C++. Объектно-ориентированное программирование: Учебное пособие / В. В. Лаптев. – СПб.: Питер, 2008. – 464 с. – ISBN 978-5-91180-200-4.
12. Керниган Б. У. Язык программирования C, 2-е издание / Б. У. Керниган, Д. М. Ритчи ; пер. с англ. В. Л. Бродового. – М. : Издательский дом "Вильямс", 2009. – 304 с. – ISBN 978-5-8459- 0891-9 (в пер.).
13. Страуструп Б. Язык программирования C++. Специальное издание / Б. Страуструп ; пер. с англ. Н. Н. Мартынова. – М.: Бином, 2011. – 1136 с. – ISBN 978-57989-0425-9 (в пер.).

Додаток А

Приклад оформлення титульного аркуша

Міністерство освіти і науки України
Вінницький національний технічний університет
Інститут інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Звіт
з лабораторної роботи №1
"Розробка програм з лінійною структурою"
Варіант № 5

Розробив студент гр. БС-14
_____ Холодов І. В.

Лабораторну роботу захищено
з оцінкою _____

Перевірив ст. викл. каф. ЗІ
__ _____ Каплун В. А.
_____ 2015 р.

Навчальне видання

Валентина Аполінаріївна Каплун
Юрій Володимирович Баришев
Аліна Василівна Остапенко

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ
Лабораторний практикум

Навчальний посібник

Редактор

Оригінал-макет підготовлено В. Каплун

Підписано до друку р.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк.
Наклад прим. Зам. № 2014-№ 2010-195

Вінницький національний технічний університет,
науково-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-81-59.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.