Міністерство освіти і науки України
Вінницький національний технічний університет

Методичні вказівки до виконання лабораторних робіт з курсу «Програмування» для студентів напряму 6.050201 «Системна інженерія»

Вінниця
ВНТУ
2015

Рекомендовано до друку Методичною радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № -- від --.--.---- р.)

Рецензенти:
П.І. Кулаков, к.т.н., доцент кафедри МПА
В.В. Ковтун, к.т.н., доцент кафедри КСУ

Методичні вказівки до виконання лабораторних робіт з курсу «Програмування» для студентів напряму 6.050201 «Системна інженерія» / Уклад. Довгалець С.М., Маслій Р.В. – Вінниця : ВНТУ, 2015. – 77 с.

У даних методичних вказівках наводяться основні рекомендації до вивчення, підготовки та проведення лаборторних занять з дисципліни «Програмування».

Зміст

Порядок виконання лабораторних та контрольних робіт

Лабораторна робота №1. Робота у текстовому редакторі

Лабораторна робота №2. Лінійна структура

Лабораторна робота №3. Оператори вибору

Лабораторна робота №4. Оператори циклу

Лабораторна робота №5. Одновимірні масиви

Лабораторна робота №6. Алгоритми пошуку та сортування

Лабораторна робота №7. Багатовимірні масиви

Лабораторна робота №8. Функції

Лабораторна робота №10. Покажчики

Лабораторна робота №11. Робота з файлами

Порядок виконання лабораторних та контрольних робіт

Лабораторні роботи складаються з декількох програм, завдань до них. Студент повинен створити, налагодити і зберегти у робочому каталозі програми в точній відповідності з приведеними в лабораторних роботах текстами програм (вихідні програми), проробити і проаналізувати результати їхньої роботи. Розробити для кожної вихідної програми самостійне завдання. Продемонструвати викладачу роботу програм.

Загальні відомості

Програма на мові Сі завжди містить одну головну функцію **main( )**. Ключове слово **void** означає, що функція **main( )** не повертає ніяких параметрів. Тіло функції завжди знаходиться в блоці програми, який обмежений відкриваючою і закриваючою дужками **{…}**.

Основні типи даних. Визначення основних (базових) типів даних здійснюється за допомогою ключових слів. Для типів із плаваючою точкою (дійсних) використовуються ключові слова **float, double, long double**, для цілих типів – **char, int, short, long** з службовими словами **signed** та **unsigned**, які вказують, що ціла змінна вважається знаковою або беззнаковою. Операція **sizeof** обчислює розмір у байтах ділянки пам'яті, яка зайнята змінною зазначеного типу або типом даних.

Оператори вводу та виводу даних. Для виводу на екран монітору використовуються функції **printf( )**, **puts( )**, **putchar( )** та ін. Функції **putchar( ), puts( )** мають тільки один параметр і виводять відповідно символи та рядки, але не забезпечують вивід числових даних. Функція **printf( )** використовується для форматованого виводу на екран усіх типів даних за допомогою специфікаторів.

Специфікатор формату починається із символу **%**, після якого стоїть символ, що вказує тип даних: **%d** – ціле число; **%ld -** довге ціле число типа **long int**; **%u** – беззнакове ціле число; **%f** – дійсне число типа **float**; **%lf** – дійсне число типа **double**; **%Lf -** довге дійсне число типа **long double**; **%e** – дійсне число в експоненціальній формі; **%c** – символ; **%s** – рядок; **%o** – цілі числа у восьмирічній системі числення; **%x** – цілі числа в 16-тирічній системі числення; **%p** – адреса змінної в 16-тирічній системі числення.

Основні функції вводу даних із клавіатури **scanf( )**, **gets( ), getchar( )**. Функція **getchar( )** не має параметрів і вводить символи**,** функція **gets( )** має один параметр і вводить рядки. Функція **scanf( )** здійснює перетворення формату за допомогою специфікаторів і може мати декілька аргументів, дозволяючи тим самим вводити значення числових, символьних змінних та рядків. Специфікатори формату аналогічні (але не цілком) тим, що використовуються функцією виводу **printf( )**.

Оператори умови. Умовний оператор **if…else** вибирає один з двох варіантів послідовності обчислень та має наступний синтаксис:

if (вираз_1) вираз_2 else вираз_3

Якщо **вираз_1** не дорівнює 0, то виконується **вираз_2**, інакше (якщо **вираз_1** дорівнює 0) виконується **вираз_3.**

Конструкція **if…else** може містити оператори вводу-виводу значень, виконання математичних операцій або виклики власних функцій, а також і іншу інструкцію **if.**

Якщо в програмі треба врахувати більш трьох можливих варіантів, доцільно використовувати оператор множинного вибору **switch**. Синтаксис цього оператора такий:

```
switch (перемикаючий_вираз)
   {     case константний_вираз_1: оператори_1;break;
         case константний_вираз_2: оператори_2;break;
                          . . .
         default: оператори_n+1;}
```

Оператор **switch** передає керування тому оператору, константний вираз якого співпадає з значеннями перемикаючого виразу. Якщо значення змінної не задовольняє умовам жодної з гілок **case,** виконується гілка, що позначена міткою **default.** Якщо ви пропустили оператор **break,** комп'ютер виконає всі оператори, які містяться у відповідній гілці **case** та інших гілках, до першого зустрінутого в тексті **break.**

Оператори циклу. Мова Сі (Сі++) має три оператори, за допомогою яких можна організувати цикли: **for**, **while**, **do…while**.

Цикл з оператором **for** частіше використовується у випадку, коли відомо точна кількість повторів, що потрібно виконати. Цей оператор має синтаксис:

for (ініціалізація_циклу; умова; список виразів) тіло_циклу

Цикл з оператором **do…while** використовується в тих випадках, коли невідомо точна кількість повторів, але тіло циклу повинно бути виконано, принаймні, один раз. Синтаксис оператору наступний:

do тіло_циклу while (умова);

Цикл з оператором **while** використовується в тому випадку, коли невідомо точне число повторів і при цьому немає необхідності, щоб тіло циклу було виконано хоча б один раз. Синтаксис оператору наступний: **while (умова) тіло_циклу**

Одномірні масиви і покажчики. Масив – це кінцева сукупність даних одного типу, що мають одне ім'я. Покажчики – це особливий вид змінних, значеннями яких служать адреси ділянок пам'яті, виділених для об'єктів (змінних) конкретних типів. При оголошенні покажчика оголошують тип змінної, на яку буде показувати покажчик.

Ім'я масиву є покажчиком-константою, значенням якого є адреса першого елемента масиву. Отже це значення не можна змінити.

Наприклад, якщо об'являється масив у вигляді **int arr[10]**, то цим визначається не тільки виділення пам'яті для десяти елементів масиву, але і для покажчика з ім'ям **arr**, значення якого дорівнює адресі першого (нульового) елементу масиву. Оскільки ім'я масиву є покажчиком, то його значення можна привласнювати іншим покажчикам. Таким чином, можливі такі конструкції:

```
int arr[10];
int *ptr=arr;
*arr=2;
arr[0]=2;
*(arr+0)=2;
*ptr=2;
ptr[0]=2;
*(ptr+0)=2;
```

Всі вони призводить до однакового результату – початковому елементу масиву привласнюється значення **2**. У даних конструкціях покажчик **ptr,** зв'язаний з масивом, не є константою, його розмір не дорівнює довжині масиву. З елементами масивів, зв'язаними з покажчиками, можна працювати за допомогою індексів і з використанням операції розіменування. Отже правильними будуть записи для доступу до елементів: **ptr[i]**, ***(ptr+i), i[ptr]**, ***(i+ptr)**.

Ініціалізація масивів. Привласнення початкових значень елементам масивів при їх оголошенні, тобто ініціалізація масивів, може здійснюватися за умовчанням та явно.

При умовчанні зовнішні і статичні масиви ініціалізуються нулями, автоматичні – довільними даними. При явній ініціалізації можливі два випадки. Перший – із явною вказівкою числа елементів і списку початкових значень, причому список початкових значень може бути коротше зазначеного розміру масиву:

```
int a[5]={2,5,7,9,1};        // Приклад із повним списком
int b[7]={5,7,9};            // Приклад із неповним списком
```

У останньому рядку інші чотири елементи масиву **b[7]** будуть мати значення або нульові (зовнішній і статичний масиви), або їхні значення будуть невизначені (автоматичний масив).

Другий випадок – без вказівки розміру масиву. Компілятор визначає число елементів масиву за списком ініціалізації:

```
int c[]={5,7,9,8}; // Масив із чотирьох елементів
```

Символьні масиви. Послідовність символів, яка знаходиться у парних подвійних лапках, у мові Сі називається рядком. Рядок відображається у пам'яті як масив елементів типу **char**, наприкінці якого міститься символ **'\0'**(іноді такий масив називають **ASCIIZ-рядком**). Адреса першого символу може використовуватися для ініціалізації масиву типу **char**, у цьому випадку ім'я масиву й адреса першого символу стають синонімами. Наприклад:

char str[5]=”name” ;

char str[ ]=”name” ;

char str[5]={‘n‘, ‘a‘, ‘m‘, ‘e‘, ‘\0‘};

Адреса першого елемента рядка може використовуватися для ініціалізації покажчика типу **char**, наприклад:

char *strptr=”name” ;

У цьому випадку змінна – покажчик **strptr** (але не покажчик-константа) одержує значення, рівне адресі символу **‘n‘**, а сам рядок розташовується в сегменті даних завантажувального модуля програми.

Масиви покажчиків. У мові Сі визначені масиви покажчиків. Прикладом визначення масиву з п'ятьох покажчиків на об'єкти типу **int** є запис: **int *array[5];** Ім'я масиву – **array**, він містить п'ять елементів типу **int ***. Вираз **(array+1)** відповідає зсуву у пам'яті на **sizeof(int *)** байтів від початку масиву.

Масиви покажчиків на рядки частіше використовуються для раціонального розміщення в пам'яті і сортування рядків із неоднаковими розмірами. Наприклад, визначимо масив покажчиків і ініціалізуємо його елементи адресами рядків:

char *name[]={”Іванов”, ”Петро”, ”Єгорович” };

При такому визначенні в пам'яті виділяється **3*sizeof(char*)** байт для покажчиків **name[0]**, **name[1]**, **name[2]**, 7 байт – для строкового літерала **”Іванов”**, 6 – для **”Петро”** і 9 – для **”Єгорович”**.

Лабораторні роботи

Лабораторна робота №1. Робота у текстовому редакторі

Мета і задачі:
Навчитися набирати та форматувати текст у текстовому редакторі.
Порядок виконання і звітування
1. Створити тестовий файл у Блокноті, і написати в ньому свою автобіографію, в якій потрібно вказати:
- дату та місце свого народження;
- інформацію про школу в якій навчалися;
- коротку інформації про свою родину;
- досягнення у навчанні (участі у олімпіадах);
- спортивні досягнення;
- кількість балів ЗНО;
- у які ВУЗи пройшли по рейтингу;

2. Зберегти тестовий файл у папці своєї бригади.
3. Створити текстовий файл у WordPad, скопіювати набраний текст з Блокноту і здійснити наступне його форматування:
- шрифт – Times New Roman;
- розмір шрифту – 14;
- вирівнювання – по лівому краю.
- виділити жирним шрифтом дату та місце народження;
- виділити курсивом кількість балів ЗНО;

4. Створити текстовий файл у Microsoft Word, скопіювати у нього відформатований текст з WordPad. Здійснити наступну зміну форматування:
- вирівнювання – по ширині.

5. Створити таблицю, в якій створити гіперпосилання на текстові файли з автобіографіями всіх студентів підгрупи. Формат таблиці вказаний на рис. 1.

ПІП	**Автобіографія**
Прізвище, Ім’я, По-батькові	Зробити гіперпосилання на ім’я документу з автобіографією кожного студенту з підгрупи

Рисунок 1 – Формат таблиці з гіперпосиланнями на текстові документи

Підсумок
Вміти здійснювати елементарне форматування тексту у текстових редакторах.

Контрольні питання

1. Як змінити шрифт у WordPad?
2. Як виділити текст курсивом, жирним чи підкресленим у WordPad ?
3. Чим відрізняється текстовий редактор від текстового процесора?
4. Для чого призначений Блокнот ?
5. Для чого призначений WordPad ?
6. Як скопіювати текст у WordPad?
7. Як вставити формулу у Microsoft Word ?
8. Які способи збереження документу у Microsoft Word Ви знаєте ?
9. Як перевірити орфографію у Microsoft Word?
10. Які способи створення таблиць у Microsoft Word Ви знаєте ?

Лабораторна робота №2. Лінійна структура

Мета і задачі:
Навчитися створювати та відлагоджувати програми з лінійною структурою на мові програмування С.
Теоретичні відомості і методичні вказівки
Алгоритм називається *лінійним*, якщо всі його дії виконуються послідовно, одна за одною, від початку до кінця. У лінійному алгоритмі не може бути команди, яка б передбачала різну послідовність виконання алгоритму.
Формально при побудові схем лінійних алгоритмів використовуються три символи (рисунок 1):

- термінатор;
- ввід/вивід;
- процес.

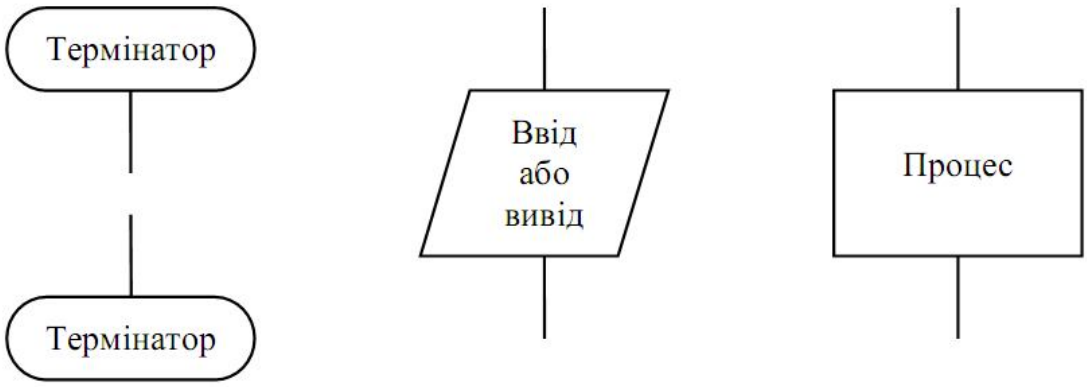


Рисунок 1 – Символи, які використовуються при побудові схем лінійних алгоритмів

Символ “Термінатор” починає і завершує будь-яку схему алгоритму. В ньому записується відповідне слово: “Початок” або “Кінець”. Від блоку “Початок” відходить лише одна лінія. Всередині символу “Ввід/вивід” записуються значення, які вводяться в алгоритм (програму) або виводяться з нього. При цьому вказується відповідне слово: “Ввід” або “Вивід”.
Всередині символу “Процес” записуються текстові вказівки, формули або оператори мови програмування. З символів “Ввід/вивід” та “Процес” може виходити лише одна лінія. За допомогою ліній, якими з’єднуються символи, позначається послідовність виконання кроків алгоритму. Після виконання операцій, розташованих в одному символі переходять по лінії до виконання операцій іншого символу.
При з’єднанні символів застосовують наступне правило, яке визначає, чи потрібно ставити стрілку на кінці з’єднувальної лінії. Якщо лінія задає так званий основний напрямок виконання операцій (зверху вниз або зліва

направо), то стрілки на кінцях ліній можна не ставити. В протилежному випадку стрілки є обов'язковими (рисунок 2).

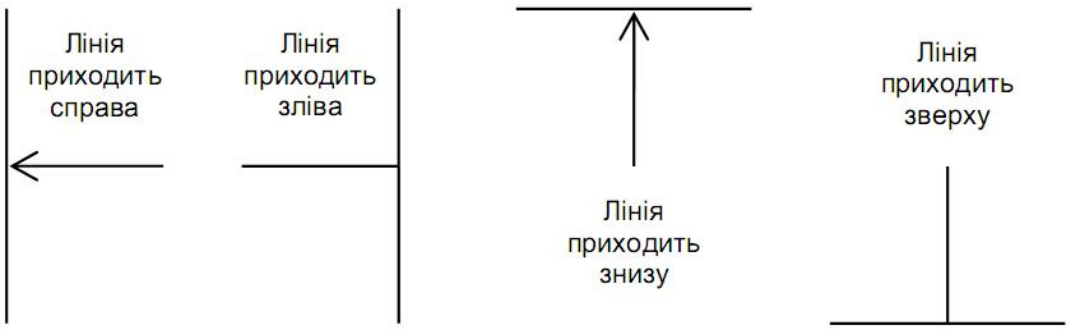


Рисунок 2 – Ілюстрація правила виставлення стрілок на кінцях з'єднувальних ліній в схемі алгоритму

Лінії, якими з'єднуються символи, не повинні перетинатись. В разі складності або неможливості (наприклад, при розміщенні схеми алгоритму на декількох сторінках) забезпечення нерозривності з'єднувальних ліній слід використовувати з'єднувальний символ, як показано на рисунку 3. В кожному з'єднувальному символі проставляється числовий ідентифікатор. Такими, що з'єднані між собою, вважаються ті з'єднувальні символи, які містять однакові ідентифікатори.

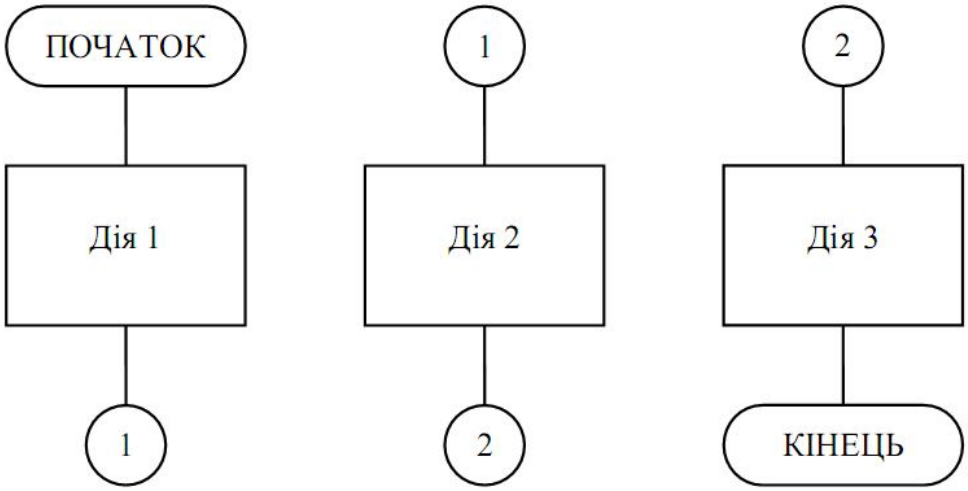


Рисунок 3 – Приклад використання з'єднувального символу

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. До кожної програми скласти схему програми.
6. Зробити висновки.

7. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, схем програм, висновків по роботі.

Варіанти завдань

Варіант 0.

1. Написати програму, яка обчислює площу трикутника, якщо відомі координати його кутів.
Вхідні дані:
Координати кутів:
x1, y1 – -2 5;
x2, y2 – 1 7;
x3, y3 – 5 -3.
Вихідні дані:
Площа трикутника – 23,56 кв. см.

2. Написати програму перерахунку величини часового інтервалу, заданого в хвилинах, в величину, виражену в годинах і хвилинах.
Вхідні дані:
Часовий інтервал – 150 хвилин.
Вихідні дані:
150 хвилин – це 20 год. 30 хв.
Підсумок
Після виконання лабораторної роботи студент повинен вміти створювати програми з лінійною структурою на мові С.

Варіант 1.

1. Написати програму обчислення площі паралелограма.
Вхідні дані:
Довжина – 9 см;
Ширина – 7,5 см;
Вихідні дані:
Площа паралелограма – 67,50 кв. см.

2. Написати програму обчислення опору електричного ланцюжка, який складається із двох паралельно з'єднаних опорів.
Вхідні дані:
Величина першого опору – 15 Ом;
Величина другого опору – 20 Ом;
Вихідні дані:
Опір ланцюжка – 8,57 Ом.

Варіант 2.

1. Написати програму обчислення об'єму паралелепіпеда.

Вхідні дані:
Довжина – 9 см;
Ширина – 7,5 см;
Висота – 5 см.
Вихідні дані:
Об’єм – 337,50 куб. см.

2. Написати програму обчислення опору електричного ланцюжка, який складається із двох послідовно з’єднаних опорів.
Вхідні дані:
Величина першого опору – 15 Ом;
Величина другого опору – 27,3 Ом;
Вихідні дані:
Опір ланцюжка – 42,30 Ом

Варіант 3.
1.Написати програму обчислення площі поверхні паралелепіпеда.
Вхідні дані:
Довжина – 9 см;
Ширина – 7,5 см;
Висота – 5 см.
Вихідні дані:
Площа поверхні – 90,00 кв. см.

2. Написати програму обчислення сили току в електричному ланцюжку.
Вхідні дані:
Напруга – 36 Вольт;
Опір – 1500 Ом;
Вихідні дані:
Сила струму – 0,024 А.

Варіант 4.
1. Написати програму обчислення об’єму куба.
Вхідні дані:
Довжина ребра – 9,5 см;
Вихідні дані:
Об’єм куба – 857,38 куб. см

2. Написати програму обчислення відстані між населеними пунктами, зображеними на карті.
Вхідні дані:
Масштаб карти – 120.
Відстані між точками, зображених населеними пунктами – 3,5 см.

Вихідні дані:
Відстань між населеними пунктами – 420 км.

Варіант 5.
1. Написати програму обчислення об’єму циліндра.
Вхідні дані:
Радіус основи – 5 см;
Висота циліндра – 10 см.
Вихідні дані:
Об’єм циліндра – 1570,80 куб. см.

2. Написати програму обчислення вартості поїздки на автомобілі на дачу (туди і назад).
Вхідні дані:
Відстань до дачі – 67 км.
Витрати бензину – 8,5 л. на 100 км.
Ціна літра бензину – 6,5 грн.
Вихідні дані:
Поїздка на дачу туди і назад – 74,04 грн.

Варіант 6.
1. Написати програму обчислення вартості покупки, яка складається з деяких зошитів та олівців.
Вхідні дані:
Ціна зошита – 2,75 грн.;
Кількість зошитів – 5;
Ціна олівця – 0,85 грн.;
Кількість олівців – 2.
Вихідні дані:
Вартість покупки – 15,45 грн.

2. Написати програму обчислення швидкості, з якою бігун пробіг дистанцію.
Вхідні дані:
Довжина дистанції – 1000 м.
Час – 3 хв. 25 сек..
Вихідні дані:
Ви пробігли зі швидкістю – 17,56 км/час

Варіант 7.
1. Написати програму обчислення вартості покупки, яка складається з деяких зошитів та такої ж кількості обгорток до них.
Вхідні дані:

Ціна зошита – 2,75 грн.;
Ціна обгортки – 0,85 грн.;
Кількість комплектів – 7 шт.
Вихідні дані:
Вартість покупки – 22,75 грн..

2. Написати програму обчислення об’єму циліндра.
Вхідні дані:
Радіус основи – 5,5 см.
Висота циліндра – 7 см.
Вихідні дані:
Об’єм циліндра – 665,23 куб. см.

Варіант 8.
1. Написати програму обчислення вартості деякої кількості (по вазі) яблук.
Вхідні дані:
Ціна одного кілограма яблук – 8,5 грн.;
Вага яблук – 2,3 кг;
Вихідні дані:
Вартість покупки – 19,55 грн.

2. Написати програму обчислення площі поверхні циліндра.
Вхідні дані:
Радіус основи – 5,5 см.
Висота циліндра – 7 см.
Вихідні дані:
Площа поверхні – 431,97 кв. см.

Варіант 9.
1. Написати програму обчислення площі трикутника, якщо відома довжина двох його сторін і величина кута між цими сторонами .
Вхідні дані:
Довжина двох сторін трикутника – 25,17 см.;
Величина кута між сторонами трикутника – 30 градусів.
Вихідні дані:
Площа трикутника – 106,25 кв. см.

2. Написати програму перерахунку відстані із верст в кілометри (1 верста – це 1066,8 м.)
Вхідні дані:
Відстань в верстах – 100.
Вихідні дані:
100 верст – це 106,8 км

Контрольні питання

1. Дайте визначення лінійного алгоритму та назвіть основні символи, які використовуються при побудові схем програм лінійних алгоритмів;
2. Перелічіть та охарактеризуйте основні числові типи даних мови С.

Лабораторна робота №3. Оператори вибору

Мета і задачі:

Навчитися створювати та відлагоджувати програми з розгалуженням обчислювальних процесів використовуючи оператори вибору на мові програмування С.

Теоретичні відомості і методичні вказівки

Мова програмування С включає три види керуючих структур (операторів), які працюють на основі вибору альтернатив: оператор **if** (конструкція **if – else**), тернарний оператор (**?:**) і оператор **switch** (конструкція **switch – case**).

*Умовний оператор **if** (конструкція **if – else**)*

Конструкція з одиночним оператором **if** призначена для виконання команди або блоку команд залежно від того, істинним або хибним є заданий логічний вираз – "умова". Загальний формат такої конструкції має наступний вигляд:

if (Вираз) true-оператор;
else false-оператор;

Якщо логічний вираз є істинним, виконуватимуться дії з переліку дій 1, якщо хибним – з переліку дій 2, тобто записані в блоці оператора **else**. Загальна схема програми оператора **if-else** представлена на рис 1.

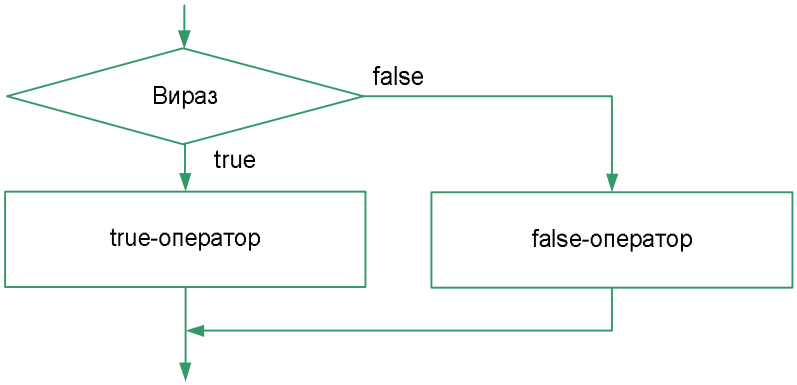


Рисунок 1 – Загальна схема програми оператора if-else

Приклад:

```
printf ("Введіть свій вік: ");
scanf ("%i", &Age);
if (Age >= 18)
{
  printf ("Ви – повнолітня людина\n");
```

```
}
else
{
  printf (“Ви – неповнолітня людина\n”);
}
```

Якщо немає необхідності вказувати перелік дій, які треба виконати в разі хибності умови, то оператор **if** записується:

if (Вираз) true-оператор;

Така конструкція відповідає схемі програми, представленій на рисунку 2.

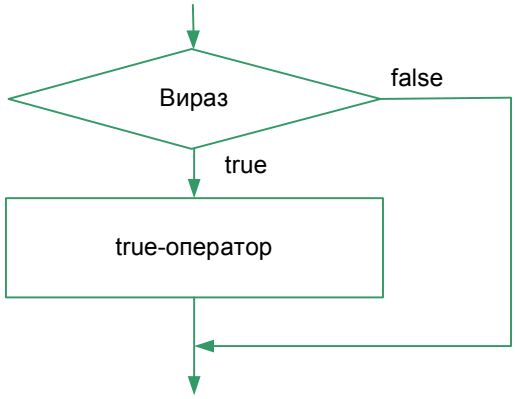


Рисунок 2 – Загальна схема програми оператора if

Логічний вираз, який обробляється оператором **if**, може набувати двох значень: “істина” і “неправда”, однак взагалі, з точки зору компілятора істинним є будь-яке ненульове значення, а хибним – нульове. Це означає, що перевірку виду:

```
if (x <> 0) ...
```

(“якщо x не дорівнює нулю”) можна замінити на більш стислу, хоча й менш наочну:

```
if (x) ...
```

В цьому випадку, якщо змінна x містить довільне число, яке відрізняється від нуля, воно буде вважатись таким, що дорівнює значенню “істина” з відповідними наслідками.
Аналогічно, умова виду:

```
if (x == 0) ...
```

може бути замінена на

```
if (!x) ...
```

(читається “якщо не-ікс”). Тоді нульове значення змінної x, яке символізує “хибність”, операцією логічного заперечення – “!” – буде перетворене в “істину” і навпаки.
Приклад:

```
printf (“Введіть стартовий капітал фірми, грн: ”);
scanf (“%i”, &StartMoney);
if (!StartMoney)
{
  printf (“Якщо капіталу немає, неможливо працювати далі\n”);
}
else
{
  ...
```

Тернарний оператор (?:)

Тернарний оператор дозволяє замінити конструкцію **if – else** більш компактним записом. Формат тернарного оператора має наступний вигляд:

(логічний вираз) ? дія 1 : дія 2;

Після перевірки значення логічного виразу в разі його істинності виконується дія 1, а в разі хибності – дія 2.

Приклад:

```
(Age >= 18) ? printf (“Повнолітній”) : printf (“Неповнолітній”);
```

Оператор множинного розгалуження swicth

В деяких задачах виникає необхідність перевірки цілочисельної змінної або виразу на рівність ряду сталих (константних) значень. Таку перевірку можна здійснити за допомогою структури з декількох конструкцій if – else, наприклад:

```
if (DayNumber == 1) printf (“Понеділок”);
else if (DayNumber == 2) printf (“Вівторок”);
else if (DayNumber == 3) printf (“Середа”);
else if (DayNumber == 4) printf (“Четвер”);
else if (DayNumber == 5) printf (“П’ятниця”);
```

```
else if (DayNumber == 6) printf (“Субота”);
else printf (“Неділя”);
```

Однак, таку структуру можна реалізувати більш компактно і наочно за допомогою оператора **switch**, який має наступний формат:

```
switch ( Вираз )
{
case константний вираз 1: набір операторів 1; break;
case константний вираз 2: набір операторів 2; break;
case константний вираз n: набір операторів n; break;
default : набір операторів по замовченню;
}
```

Конструкція **switch – case** працює наступним чином. Цілочисельний вираз послідовно порівнюється із константним значенням 1, 2 і т. д. Якщо деяке значення k співпадає із значенням виразу, то виконується відповідна дія k, після чого оператор **break** передає управління за межі блоку **switch**. Якщо серед констант не знайшлось рівної виразу, то управління передається на мітку **default**. За відсутності цієї мітки управління передається за межі блоку **switch**.
На рис. 3 представлена загальна схема програми оператора множинного розгалуження **switch**.

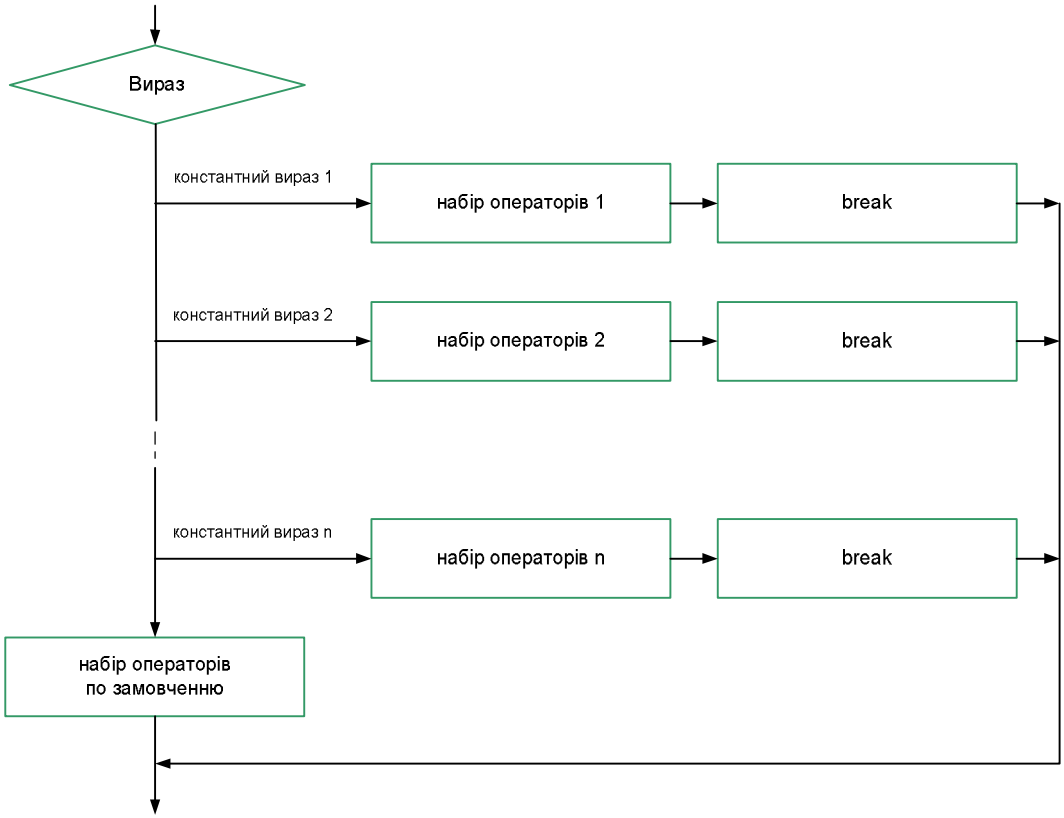


Рисунок 3 – Загальна схема програми оператора множинного розгалуження switch

Таким чином, реалізація перевірки з прикладу про дні тижня на основі оператора **switch** матиме наступний вигляд:

```
swicth (DayNumber)
{
 case 1: printf (“Понеділок”); break;
 case 2: printf (“Вівторок”); break;
 case 3: printf (“Середа”); break;
 case 4: printf (“Четвер”); break;
 case 5: printf (“П’ятниця”); break;
 case 6: printf (“Субота”); break;
 default: printf (“Неділя”);
}
```

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. До кожної програми скласти схему програми.
6. Зробити висновки.

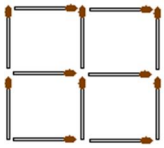
7. Звіт по лабораторній роботі має складатися з титульної сторінки, схем програм, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 0.
Написати програму, яка запитує в користувача номер дня (1-30) червня поточного року і виводить відповідну номеру назву дня тижня.

Варіант 1.
Написати програму, яка запитує в користувача номер дня тижня і виводить відповідну номеру назву дня тижня.

Варіант 2.



Написати програму, яка обчислює скільки сірників потрібно щоб скласти «ґрати» N x N, у діапазоні 0<N<200.
Наприклад при N=2 відповідь 12.
Перевіряти відповідність вхідних даних заданому діапазону.

Варіант 3.
Написати програму перевірки знання дати початку другої світової війни, у випадку правильної відповіді виводити напис «правильно», у випадку неправильної відповіді програма повинна робити підказку користувачеві, і тільки після п'яти неправильних відповідей виводити правильну відповідь.

Варіант 4.
Написати програму обчислення вартості покупки з урахуванням знижки. Знижка в 3% надається якщо вартість покупки більше 500 гривень, в 5% якщо вартість більше 1000 гривень.

Варіант 5.
Написати програму обчислення вартості катання на ковзанах у льодовому комплексі. Вхідні дані - час перебування на ковзанці, вид картки (VIP-картка - 20% знижка від базової вартості за годину, Premium-картка - 10% знижка від базової вартості за годину, Base-картка - базова вартість за годину).

Варіант 6.

Написати програму, яка запитує в користувача ступінь двійки, і виводить на екран число в цьому ступені. Якщо введено негативне число виводиться повідомлення про помилку.

Варіант 7.
Написати програму, яка запитує в користувача чотиризначне число. Знайти частку перших двох цифр цього числа розділених на різницю останніх двох. У випадку введення не чотиризначного числа вивести повідомлення про помилку.

Варіант 8.
Написати програму, яка після введеного із клавіатури числа (у діапазоні від 1 до 99), що позначає грошову одиницю, дописує слова «копійка» у правильній формі. Наприклад 5 копійок, 3 копійки тощо.

Варіант 9.
Написати програму, яка запитує в користувача п'ятизначне число. Знайти добуток останніх трьох цифр цього числа розділених на суму перших двох. У випадку введення не п'ятизначного числа вивести повідомлення про помилку.

Після виконання лабораторної роботи студент повинен:
1. Знати таблиці істинності для логічних операцій та побітових логічних операцій.
2. Вміти використовувати логічні операції для написання програм з розгалуженням обчислювальних процесів на мові програмування С.
3. Вміти створювати програми з розгалуженням обчислювальних процесів використовуючи оператори вибору **if**, **if-else**, **switch** на мові програмування С.

Контрольні питання
1. Які оператори вибору існують в мові С ?
2. Яку роль відіграє оператор **if** при написанні програм ?
3. Який тип виразу має оператор умови ?
4. Як виконується робота оператора **if** ?
5. Чи обов’язково повинні бути присутні всі гілки оператора **if** ?
6. Чи допускається вкладеність операторів **if** ?
7. Яка глибина вкладень допускається ?
8. Скільки операторів може містити кожна з гілок оператора **if** ?
9. Що потрібно зробити, коли в кожній гілці треба виконати не один, а більше операторів?
10. Яке правило застосовується щоб уникнути неоднозначності використання зарезервованого слова **else**?

11. Яку роль виконує оператор **switch** ?
12. Який тип виразу має “селектор” оператора **switch ?**
13. Скільки міток може містити кожна з гілок оператора **switch** ?
14. Як виконується робота оператора **switch** ?
15. Коли ефективно використовувати оператор **switch** ?
16. Скільки операторів може мати кожна з гілок оператора **switch** ?
17. Якого типу можуть бути мітки в операторі **switch** ?
18. Оператори вибору використовують оператори відношення. Які оператори відношення Ви знаєте ?
19. Які типи операндів допустимі в операторах відношення, і який результат будуть мати ці відношення ?
20. Якого типу операнди можуть бути використані в операторах відношення ?
21. Яким чином виконується порівняння рядкових даних ?
22. Які логічні операції виконуються в мові С ?
23. Який пріоритет мають логічні операції порівняно з операціями відношення ?
24. Чи мають місце якісь особливості при виконанні логічних операцій ?
25. Яка різниця між моделями повного обчислення логічних виразів і за короткою схемою?
26. В яких випадках доречна коротка схема обчислення логічних виразів ?
27. В яких випадках доречна повна схема обчислення логічних виразів ?
28. Який пріоритет мають логічні операції порівняно з арифметичними операціями?
29. В основі яких алгоритмів лежать оператори відношення ?
30. Побудуйте таблиці істинності логічних функцій НІ, АБО, ТА.

Контрольні вправи

1. З допомогою операторів вибору і присвоювання записати фрагмент програми, який обчислює значення змінної *n* по наступному правилу:

$$n = \begin{cases} n+1 \text{ при } i = 1 \text{ або } i = 5 \\ a+b \text{ при } i = 7 \text{ або } i = 12 \\ a-b \text{ в нших випадках} \end{cases}$$

2. З допомогою операторів вибору і присвоювання записати фрагмент програми, який обчислює значення змінної *n* по наступному правилу

$$n = \begin{cases} n+1 \text{ при } a > 0 \text{ і } b = 0 \\ a+b \text{ при } a <= 0 \text{ і } b = 0 \\ a-b \text{ в нших випадках} \end{cases}$$

3. З допомогою операторів вибору і присвоювання записати фрагмент програми, який обчислює значення змінної *n* по наступному правилу:

$$n = \begin{cases} 1 \text{ при i} = 1 \text{ або } 2 \text{ або } 7 \\ 2 \text{ при i} = 10 \\ 0 \text{ в інших випадках} \end{cases}$$

4. Зобразити фрагмент схеми програми, який відповідає наступному фрагменту програми:

```
if ( c<3 )  if ( c == 2)   a++;  else b++; a += 1;
```

5. Зобразити фрагмент схеми програми, який відповідає наступному фрагменту програми:

```
if ( c<3 )  if ( c == 2)   a++;  b++; a += 1;
```

6. Зобразити фрагмент схеми програми, який відповідає наступному фрагменту програми:

```
if ( c<3 )  if ( c == 2)   a++;  else b++; if ( c<2 ) c++;   a += 1;
```

7. Зобразити фрагмент схеми програми, який відповідає наступному фрагменту програми:

```
if ( c<3 )  if ( c == 2)   a++;  else b++; if ( c<2 ) c++; else  a += 1; { c++; b++;  }
```

8. Ввести і надрукувати значення елементів масиву дійсного типу з розміром 10. Обчислити і надрукувати кількість негативних елементів масиву.
9. Ввести і надрукувати значення елементів масиву дійсного типу з розміром 20. Обчислити і надрукувати індекс найменшого елементу масиву.
10. Ввести і надрукувати значення елементів масиву цілого типу з розміром 20. Обчислити і надрукувати середнє арифметичне для елементів масиву. Постарайтесь, щоб дробова частина в результаті не загубилась.
11. Ввести і надрукувати значення змінних a, b, c дійсного типу. Обчислити і надрукувати, скільки поміж них відмінних від нуля.
12. Ввести і надрукувати значення змінних a, b цілого типу. Обчислити, чи рівні вони один одному, і надрукувати відповідь.
13. Ввести і надрукувати значення змінних a, b, c дійсного типу. Обчислити кількість позитивних значень серед заданих і надрукувати відповідь.

14. Ввести і надрукувати значення змінних x дійсного типу. Обчислити і надрукувати, значення функції y =| x |.
15. Вивести і надрукувати значення x дійсного типу. Обчислити і надрукувати значення y:

$$n = \begin{cases} +1 & \text{при } x > 0 \\ 0 & \text{при } x = 0 \\ -1 & \text{при } x < 0 \end{cases}$$

Лабораторна робота №4. Оператори циклу

Мета і задачі:

Навчитися створювати та відлагоджувати програми з циклічними обчислювальними процесами використовуючи оператори циклу на мові програмування С.

Теоретичні відомості і методичні вказівки

Циклічні структури (цикли) дозволяють програмісту визначити дії, які необхідно повторювати, поки деяка умова залишається істиною. Мова С забезпечує три типи циклічних структур, які будуються на операторах **for**, **while** і **do – while**, а також два оператори, призначені для зміни послідовності виконання циклу: **break** і **continue**.

Оператор циклу for

Оператор for зазвичай використовується для організації циклу, який керується лічильником (циклу з параметром). Однак формат цього оператора в мові С є надзвичайно гнучким і має наступний вигляд:

```
for (вираз 1; вираз 2; вираз 3)
{
 тіло циклу;
}
```

Оператор працює таким чином, що "вираз 1" буде виконано лише один раз в момент входу до циклу, "вираз 3" буде виконуватись в кінці кожного кроку циклу, а сам цикл буде тривати до тих пір, поки "вираз 2" має значення "істина" (тобто взагалі будь-яке, відзначне від нуля). При цьому перевірка значення "виразу 2" здійснюється перед виконанням тіла циклу. Це означає, що цикл на основі оператора **for** є циклом з передумовою, тобто циклом "ПОКИ".

Загальна схема програми оператора циклу **for** представлена на рис. 1.

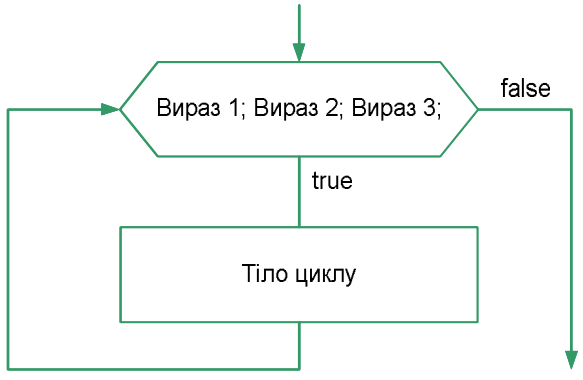


Рисунок 1 – Загальна схема програми оператора циклу for

Гнучкість оператора **for** в мові програмування С дозволяє будувати на його основі навіть такі конструкції, які на перший погляд можуть здаватись безглуздими, наприклад:

```
for (;;);
```

що означає нескінченний цикл.
Зрозуміло, що подібні "екзотичні" конструкції застосовуються доволі рідко. На практиці використовується наступний формат оператора **for**:

```
for (ініціалізація лічильника; перевірка лічильника; зміна лічильника)
{
 тіло циклу;
}
```

Приклад:

```
S = 0.0;
k = 1;
// цикл, в якому змінна-лічильник i
// буде послідовно набувати значення 0, 1, 2, ..., n – 1
for (i = 0; i < n; i++)
{
 S += pow (x, i) * sin (x) / k;
 k *= (i + 1);
}
```

В наведеному прикладі останнє значення, яке набуде змінна-лічильник i, дорівнює n – 1. Відразу після набуття лічильником значення n вираз "i < n" стане хибним і цикл припиниться.

Цикловий оператор while
Оператор **while** як і оператор **for** призначений для організації циклів з передумовою і має наступний формат:

```
while (вираз)
{
 тіло циклу;
}
```

На відміну від оператора **for**, оператор **while** автоматично виконує лише одну операцію перед кожним кроком циклу: перевіряє значення

виразу, розміщеного в дужках і передає управління залежно від результату перевірки.
Цикл триває до тих пір, поки вираз має істинне значення.

Приклад:

```
S = 0.0;
k = 1;
i = 0;
// ПОКИ i < n цикл триватиме
while (i < n)
{
  S += pow (x, i) * sin (x) / k;
  k *= ++i;
}
```

Загальна схема програми оператору циклу **while** зображена на рис. 2.

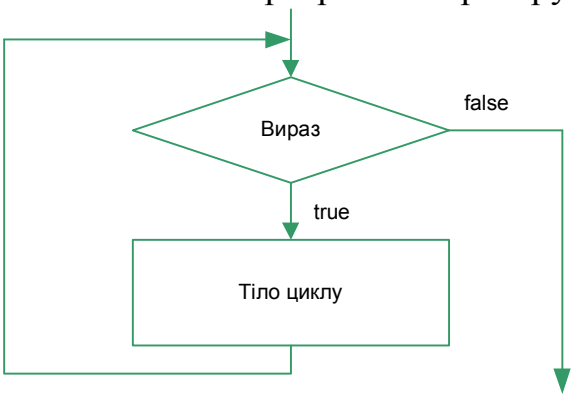


Рисунок 2 – Загальна схема програми оператора циклу while

Циклова конструкція do-while

Принцип роботи конструкції **do – while** аналогічний оператору **while**, однак ця конструкція призначена для організації циклів з післяумовою (циклів “ДО”) і має наступний формат:

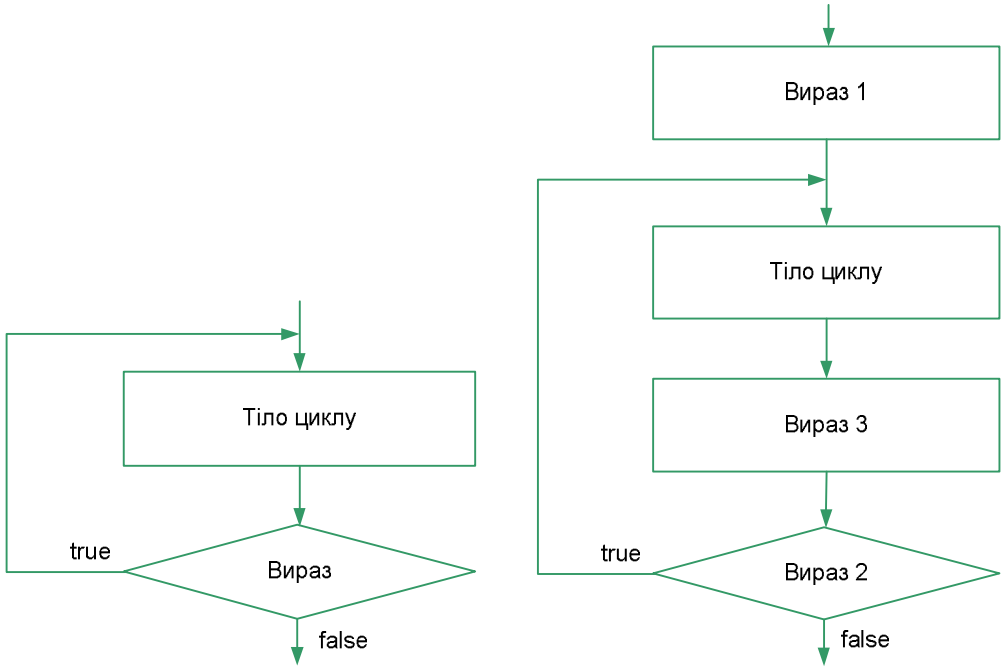
```
do {
  тіло циклу;
} while (вираз);
```

Приклад:

```
S = 0.0;
k = 1;
i = 0;
```

```
// тіло циклу повторюватиметься ДО тих пір...
do {
  S += pow (x, i) * sin (x) / k;
  k *= (i++ + 1);
// поки i < n
} while (i < n);
```

Загальна схема та розгорнуті схеми програм оператору циклу **do - while** зображені на рис. 3.



а) б)

Рисунок 3 – Схеми програм оператора циклу do-while

На рисунку 3а) зображена загальна схема програми оператора циклу **while**, схему програми на рисунку 3б) можна інтерпретувати як розгорнуту схему програми оператора **for**.

Оператор break

Оператор **break** призначений для примусового переривання циклу. В наступному прикладі реалізується обчислення суми чисел, які вводяться користувачем аж до вводу першого нуля.
Приклад:

```
S = 0;
// ненульове значення вважається істиною,
// тому даний цикл є безкінечним
while (1)
```

```
{
  printf (“Введіть чергове число (нуль – завершення): ”);
  scanf (“%i”, &num);
// якщо введено нуль – перервати цикл
  if (!num) break;
  S += num;
}
printf (“Сума введених чисел дорівнює %i”, S);
```

Наведений приклад має суто ілюстративне значення, тому що поставлену задачу можна розв’язати і більш компактним шляхом, наприклад:

```
S = 0;
do {
  printf (“Введіть чергове число (нуль – завершення): ”);
  scanf (“%i”, &num);
  S += num;
} while (num);
printf (“Сума введених чисел дорівнює %i”, S);
```

Оператор continue

Оператор **continue** призначений для примусового (позачергового) переходу до наступного кроку циклу. В наступному прикладі реалізується обчислення суми десяти чисел, які вводяться користувачем, за виключенням від’ємних.

Приклад:

```
S = 0;
for (int i = 0; i < 10; i++)
{
  printf (“Введіть %i-те число: ”, i);
  scanf (“%i”, &num);
  // якщо введено від’ємне число –
  // перейти до наступного кроку циклу
  if (num < 0) continue;
  S += num;
}
printf (“Сума введених чисел дорівнює %i”, S);
```

Наведений приклад має суто ілюстративне значення, тому що поставлену задачу можна розв’язати і більш компактним шляхом, наприклад:

```
S = 0;
```

```
for (int i = 0; i < 10; i++)
{
  printf (“Введіть %i-те число: ”, i);
  scanf (“%i”, &num);
  if (num >= 0) S += num;
}
printf (“Сума введених чисел дорівнює %i”, S);
```

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. До кожної програми скласти схему програми.
6. Зробити висновки.
7. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 0.
Написати програму, яка перетворює введене користувачем, десяткове число у двійкове, діапазон у десятковому еквіваленті від 1 до 256.

Варіант 1.
Написати програму, яка обчислює і виводить на екран середнє значення для декількох цілих чисел. Нехай останнім значенням, яке завершує введення буде контрольне число 9999.

Варіант 2.
Написати програму, яка виводить на екран фігуру у вигляді ромба, що складається із зірочок. Кількість рядків у ромбі задавати із клавіатури. Наприклад:

```
   *
  ***
******
  ***
   *
```

Варіант 3.
Написати програму, яка перетворює введене користувачем двійкове число в шістнадцяткове, діапазон у десятковому еквіваленті від 1 до 256.

Варіант 4.
Написати програму, яка виводить таблицю значень функції $y=-2,4x^2+5x-3$ у діапазоні від –2 до 2, із кроком 0,5.

Варіант 5.
Написати програму, яка обчислює суму перших членів ряду: 1+1/2+1/3+1/4+...Кількість доданих членів ряду задається під час роботи програми.

Варіант 6.
Програма повинна зчитувати числа з клавіатури, перше зчитувальне число має визначати кількість зчитувальних чисел. Перше зчитувальне число повинно бути в діапазоні від 1 до 50, наступні зчитувальні числа повинні бути в діапазоні від 1 до 30. У випадку якщо введене число не відповідає припустимим діапазонам виводити повідомлення про помилку. Для кожного зчитувального числа програма повинна вивести рядок з рівною цьому числу кількістю зірочок.

Варіант 7.
Написати програму, яка обчислює число «Пі» із заданою користувачем точністю. Для обчислення числа «Пі» потрібно скористатися частковою сумою ряду 4-4/3+4/5-4/7+4/9-4/11+... . Точність обчислення визначається кількістю членів ряду.

Варіант 8.
Написати програму перевірки знання таблиці множення. Програма повинна вивести 10 прикладів і виставити оцінку: за 10 правильних відповідей - «відмінно», за 9 і 8 - «добре», за 7 і 6 – «задовільно», за 6 і менше - «не задовільно».

Варіант 9.
Написати програму, яка обчислює й виводить суму парних цілих чисел від 2 до 30.

Підсумок
Після виконання лабораторної роботи студент повинен вміти:
1. Використовувати логічні операції для написання програм з розгалуженням обчислювальних процесів на мові програмування С.
2. Створювати програми з циклічними обчислювальними процесами використовуючи оператори циклу **for**, **while**, **do while** на мові програмування С.

Контрольні питання

1. Що називається циклом ?
2. Для чого потрібний лічильник циклу ?
3. Загальний формат оператора **for** ?
4. Як виконується робота оператора **while** ?
5. Яку роль виконує оператор **do while** ?
6. Яка відмінність між **while** та **do while** ?
7. Як виконується робота оператора **break** ?
8. Як виконується робота оператора **continue** ?
9. Як називаються такі логічні операції як &&, ||, ! ?
10. Загальний формат оператора **while** ?
11. Загальний формат оператора do **while.**
12. Яким чином використовуються логічні операції такі як &&, ||, ! в операторах **for**, **do while**, **while**.

Контрольні вправи

1. Знайти помилку в кожному із наступних фрагментів коду:
    a. Виведення у циклі на екран значень лічильника:

```
for ( x = 100, x >=1, x++ )
printf (“%d\n”, x );
```

    b. Наступний код повинен виводити, чи є дане ціле число парним або непарним:

```
swich ( value % 2 ) {
case 0: printf (“Even integer\n”);
case 1: printf (“Odd integer\n”);
}
```

    c. Наступний код повинен виводити непарним ціле число від 999 до 1:

```
for ( x = 999; x>= 1; x += 2 )
printf (“%d\n”, x ) ;
```

    d. Наступний код повинен додавати цілі числа від 100 до 150:

```
for ( x = 100; x <= 150; x++ )
total += x ;
```

2. Напишіть оператори for, які виводять наступні послідовності значень:
    a. 1, 2, 3, 4, 5, 6, 7

b. 3, 8, 13, 18, 23

c. 20, 14, 8, 2, -4, -10

d. 19, 27, 35, 43, 51

3. Дано наступне визначення:

```
int  k;
```

При яких вихідних значеннях k наведений нижче цикл буде виконуватися безкінечно:

```
do
{
   k++;
} while ( k > -5 );
```

Можливі варіанти відповідей : при k<=…, або при k>=…., або таких k не існує.

a. `int k;`

При яких вихідних значеннях к наведений нижче цикл буде виконуватися безкінечно:

```
while ( k < 12);
```

Можливі варіанти відповідей : при k<=…, або при k>=…., або таких k не існує

4. Скільки разів буде виконано тіло приведеного нижче циклу ?

```
for ( int k=4; k<17; k+=3 );
```

Можливі варіанти відповідей: тіло циклу буде виконано n раз або цикл буде виконуватися безкінечно.

5. Нехай дані змінні:

```
int   k, n ;
```

Вкажіть, що надрукує наступний фрагмент програми (нижче знак ^ означає пробіл):

```
printf ( “\n%3s\n ”, “-” );
for ( k = 5; k > 5; k-- )
{
   n = 6 – k; printf ( “%I^^%d^%2s^^”, k, n, “-” );
}
```

6. Нехай дані змінні:

```
int  k, n ;
```

Вкажіть, що надрукує наступний фрагмент програми (нижче знак ^ означає пробіл):

```
printf ( “\n%-3.2s\n ”, “*****” );
for ( k = 5; k > 5; k-- )
{
   n = 6 – k; printf ( “%I^^%0.4d^%2s^^”, k, n, “-” );
}
```

7. Скільки разів буде виконано тіло приведеного нижче циклу.

```
int  c = 3;
for ( int  k=4; k<17; k+=3, c+=2 );
```

Яке значення буде мати змінні *c* після виходу із циклу?

8. Зобразити фрагмент схеми програми, який відповідає вправі 1.b.
9. Зобразити фрагмент схеми програми, який відповідає вправі 1.c.
10. Зобразити фрагмент схеми програми, який відповідає вправі 2.a.
11. Зобразити фрагмент схеми програми, який відповідає вправі 2.d.
12. Зобразити фрагмент схеми програми, який відповідає вправі 2.e.

Лабораторна робота №5. Одновимірні масиви

Мета і задачі:

Навчитися створювати та відлагоджувати програми, в яких використовуються статичні одновимірні масиви на мові програмування С.

Теоретичні відомості і методичні вказівки

Масив – це послідовна група комірок пам'яті, які мають однакову назву та однаковий тип. Для посилання на окрему комірку (елемент масиву), необхідно вказати назву масиву та номер позиції (індекс) цієї комірки.

Властивості масивів:

- в масиві зберігаються окремі значення, які називаються елементами;
- всі елементи масиву мають один тип;
- всі елементи масиву зберігаються в пам'яті послідовно і перший елемент має нульове зміщення адреси, тобто нульовий індекс (рисунок 9.1);
- назва масиву є константою і містить адресу нульового елементу масиву.

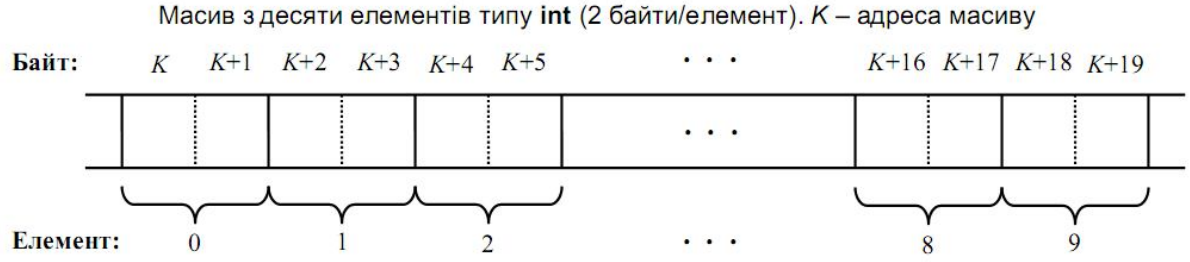


Рисунок 1 - Ілюстрація принципу розташування масиву в пам'яті

Оголошення та ініціалізація масивів

Оголошення масивів у загальному вигляді здійснюється за наступним форматом:

<тип даних> <назва> [розмірність];

Приклади:

```
/* масив чисел типу int на 150 елементів */
int Balls[150];
/* масив чисел типу float на 8 елементів */
float Value[8];
```

Ініціалізація масиву при оголошенні може бути виконана за допомогою так званого списку ініціалізаторів, наприклад:

```
int X[5] = {2, -1, 8, 100, 45};
```

Обробка масивів

Зазвичай обробка масивів здійснюється у циклах; при цьому верхня і/або нижня границя циклу певним чином "прив'язується" до кількості елементів у масиві.

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
- оголосити одновимірний статичний масив з десяти елементів типу int;
- заповнити масив з клавіатури;
- вивести результати обчислень на екран.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. Зробити висновки.
6. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 0.

Написати програму, яка обчислює:

1) мінімальний по модулю елемент масиву;

2) суму модулів елементів масиву, розташованих після першого елементу, що дорівнює нулю.

Варіант 1.

Написати програму, яка обчислює:

1) суму від'ємних елементів масиву;

2) добуток елементів масиву, розташованих між максимальним і мінімальним елементами.

Варіант 2.

Написати програму, яка обчислює:

1) суму додатних елементів масиву;

2) добуток елементів масиву, розташованих між максимальним і мінімальним по модулю елементами.

Варіант 3.
Написати програму, яка обчислює:
1) добуток елементів масиву з парними номерами;
2) суму елементів масиву, розташованих між першим і останнім нульовими елементами.

Варіант 4.
Написати програму, яка обчислює:
1) суму елементів масиву з непарними номерами;
2) суму елементів масиву, розташованих між першим і останнім від'ємними елементами.

Варіант 5.
Написати програму, яка обчислює:
1) максимальний елемент масиву;
2) суму елементів масиву, розташованих до останнього додатного елемента .

Варіант 6.
Написати програму, яка обчислює:
1) мінімальний елемент масиву;
2) суму елементів масиву, розташованих між першим і останнім додатними елементами.

Варіант 7.
Написати програму, яка обчислює:
1) номер максимального елементу масиву;
2) добуток елементів масиву, розташованих між першим і другим нульовими елементами.

Варіант 8.
Написати програму, яка обчислює:
1) номер мінімального елементу масиву;
2) суму елементів масиву, розташованих між першим і другим від'ємними елементами.

Варіант 9.
Написати програму, яка обчислює:
1) максимальний по модулю елемент масиву;
2) суму елементів масиву, розташованих між першим і другим додатними елементами.

Підсумок

Після виконання лабораторної роботи студент повинен вміти створювати програми, які оброблюють дані використовуючи статичні одновимірні масиви на мові програмування С.

Контрольні питання

1. Що таке масив?
2. Що таке одновимірний масив?
3. Які є правила оформлення масивів?
4. Для чого використовується масив типу **char**?
5. Якими способами можна ініціалізувати елементи масивів?
6. Коли і для чого використовується ключове слово **static**?
7. Як звернутися до елементу масиву ?
8. Як за допомогою функції **scanf** ввести з клавіатури масив символів ?
9. Як ініціалізувати масив при об'явленні ?
10. В чому різниця між посиланням на сьомий елемент масиву і на елемент масиву номер сім?

Контрольні вправи

1. Написати програму яка визначає номер мінімального елементу масиву.
2. Написати програму яка обчислює суму елементів масиву, розташованих між першим та другим від'ємними елементами.
3. Написати програму яка перетворює масив таким чином, щоб спочатку розташовувалися всі елементи, модуль яких не перевищує 1, а потім всі інші.
4. Написати програму яка обчислює суму додатних елементів масиву.
5. Написати програму яка обчислює добуток елементів масиву, розташованих між максимальним за модулем і максимальним за модулем елементами.
6. Написати програму яка упорядковує елементи масиву за зменшенням.
7. Написати програму яка обчислює добуток елементів масиву з парними номерами.
8. Написати програму яка обчислює суму елементів масиву, розташованих між першим та останнім нульовими елементами.
9. Написати програму яка претворює масив таким чином, щоб спочатку розташовувалися всі додатні елементи, а потім всі від'ємні елементи (елементи рівні 0 вважати позитивними).
10. Написати програму яка обчислює суму елементів масиву з непарними номерами.
11. Написати програму яка обчислює суму елементів масиву, розташованих між першим та останнім від'ємними елементами.

12. Написати програму яка стискає масив шляхом видалення з нього всіх елементи, модуль яких не перевищує 1; елементи, що звільнились в кінці вектора, заповнити нулями.
13. Написати програму яка знаходить максимальний елемент масиву.
14. Написати програму яка обчислює суму елементів масиву, розташованих до останнього позитивного елементу.
15. Написати програму яка стиснути масив шляхом видалення з нього всіх елементи, модуль яких знаходиться в інтервалі [a,b]; елементи, що звільнилися в кінці вектора, заповнити нулями.
16. Написати програму яка знаходить мінімальний елемент масиву.
17. Написати програму яка обчислює суму елементів масиву, розташованих між першим та другим позитивними елементами.
18. Написати програму яка перетворює масив таким чином, щоб спочатку розташовувалися всі елементи, які дорівнюють 0, а потім всі інші.
19. Написати програму яка визначає номер максимального елементу масиву.
20. Написати програму яка обчислює добуток елементів масиву, розташованих між першим і другим нульовими елементами.
21. Написати схему програми з вправи № 1.
22. Написати схему програми з вправи № 4.
23. Написати схему програми з вправи № 10.
24. Написати схему програми з вправи № 14.
25. Написати схему програми з вправи № 16.
26. Написати схему програми з вправи № 19.
27. Задано масив: **double** mas [50]; Написати фрагмент програми, яка надрукує з нової стрічки значення елементів масиву по чотири елемента в стрічці і по двадцять позицій на елемент. Значення, які надрукуються прижимати до правої границі поля виводу, а в дробовій частині друкувати 6 цифр. Вирішити задачу з допомогою циклу *do..while.*

Лабораторна робота №6. Алгоритми пошуку та сортування

Мета і задачі:
Навчитися створювати та відлагоджувати програми, в яких використовуються алгоритми пошуку та сортування на мові програмування С.

Теоретичні відомості і методичні вказівки
Алгоритм сортування – це алгоритм для впорядкування елементів у списку (масиві). У випадку, коли елемент списку має кілька полів, поле, що служить критерієм порядку, називається ключем сортування. На практиці у якості ключа часто виступає число, а в інших полях зберігаються будь-які дані, які ніяк не впливають на роботу алгоритму.

Оцінка алгоритмів сортування
Алгоритми сортування оцінюються за швидкістю виконання та ефективності використання пам'яті:

- Час- основний параметр, що характеризує швидкодію алгоритму. Називається також обчислювальною складністю. Для впорядкування важливі найгірша, середня та найкраща поведінка алгоритму в термінах потужності вхідної множини А. Якщо на вхід алгоритму подається множина А, то позначимо n=|A|. Для типового алгоритму хороша поведінка - це $O(n\log n)$, а погана поведінка – це $O(n^2)$. Ідеальна поведінка для упорядкування - $O(n)$.
- Пам'ять – ряд алгоритмів вимагає виділення додаткової пам'яті під тимчасове зберігання даних. Як правило, ці алгоритми вимагають $O(\log n)$ пам'яті. При оцінці не враховується місце, яке займає вихідний масив і незалежні від вхідної послідовності витрати, наприклад, на зберігання коду програми (так як все це споживає $O(1)$). Алгоритми сортування, не споживають додаткової пам'яті, відносять до сортувань на місці.

Бульбашкове сортування – простий алгоритм сортування. Для розуміння і реалізації цей алгоритм найпростіший, але ефективний для невеликих масивів. Складність алгоритму - $O(n^2)$.
Схема програми бульбашкового сортування представлена на рис. 1.

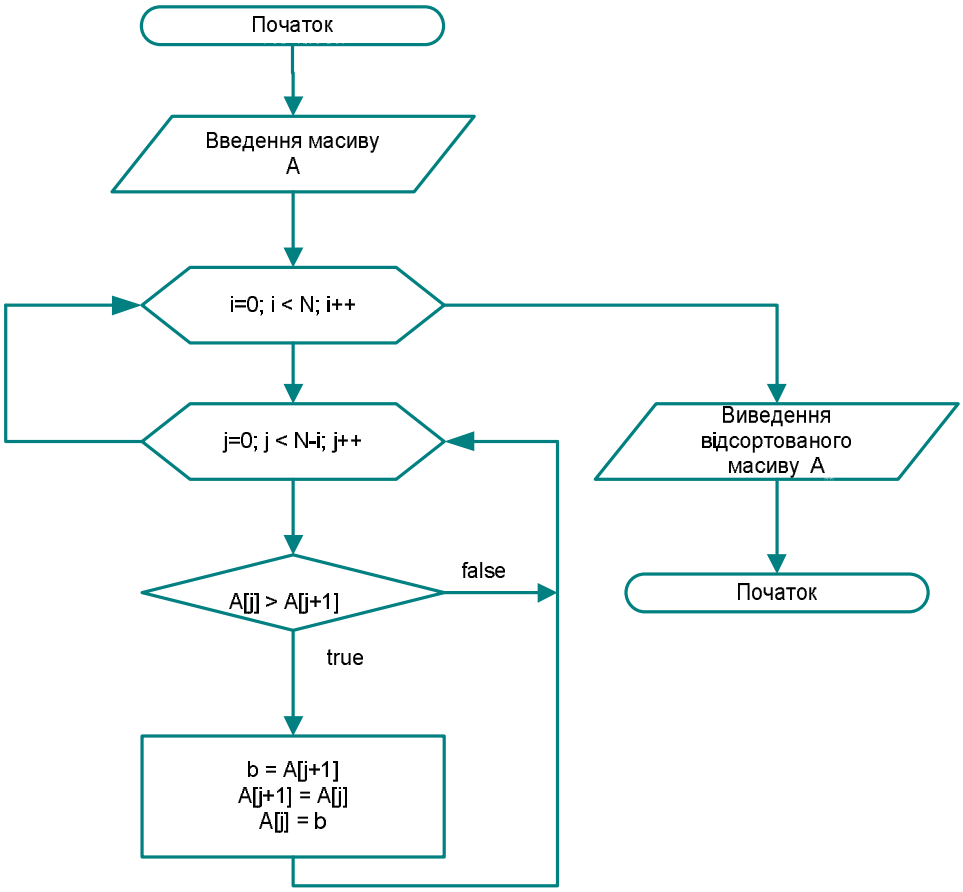


Рисунок 1 – Схема програми бульбашкового сортування

Сортування вибором – алгоритм сортування, який відноситься до нестійких алгоритмів сортування. На масиві з n елементів має час виконання в кращому, середньому і найгіршому випадках - $O(n^2)$, припускаючи що порівняння виконуються за постійний час.
Схема програми сортування вибором представлена на рис. 2.

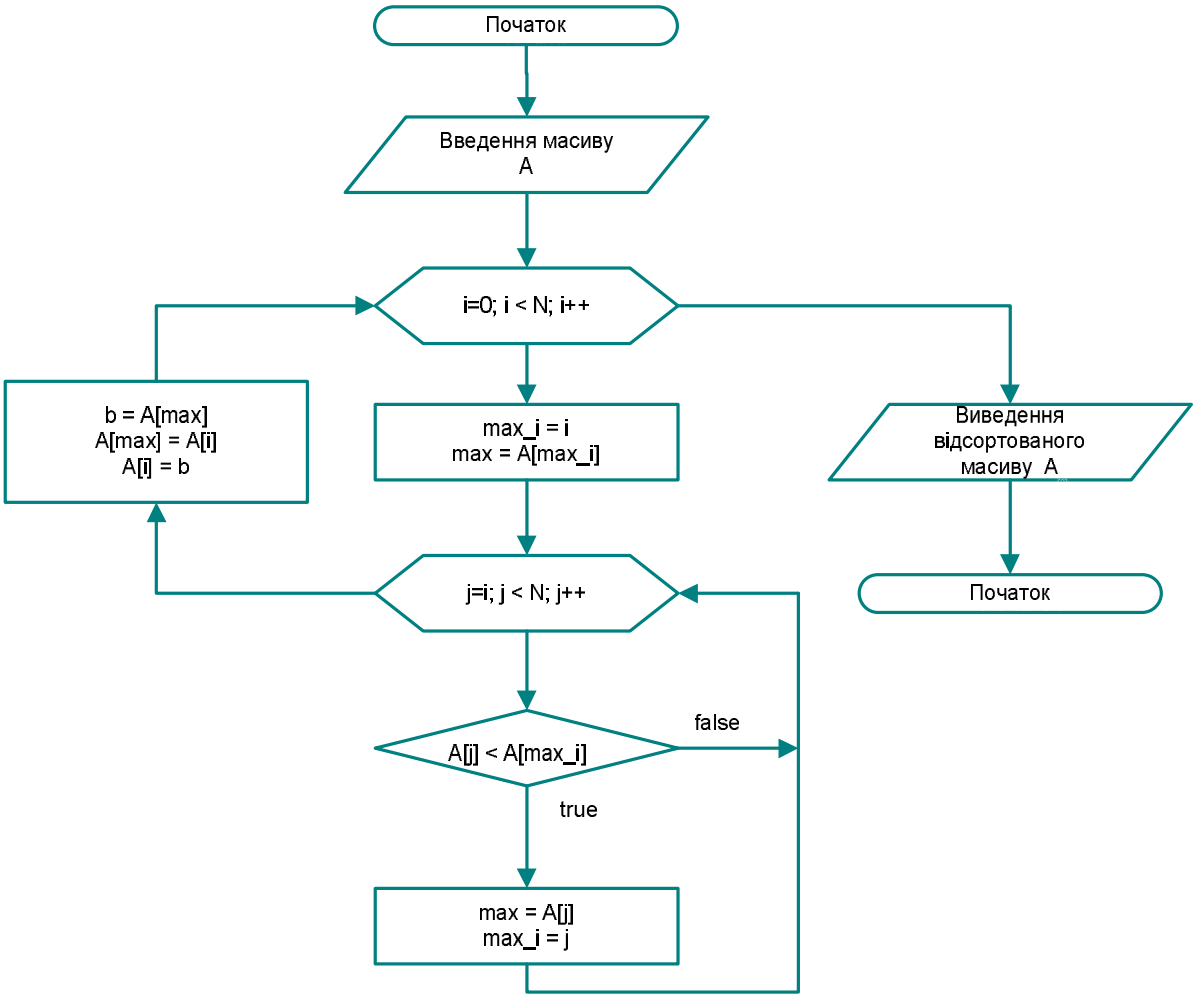


Рисунок 2 – Схема програми сортування вибором

Сортування вставками – простий алгоритм сортування. Хоча цей алгоритм сортування в ефективності більш складнішим в нього є ряд переваг:

- ефективний на невеликих наборах даних, на наборах до десятків елементів може виявитися найкращим;
- ефективний на наборах даних які вже частково відсортовані;
- це стійкий алгоритм сортування (не змінює порядок елементів які вже відсортовані);
- може сортувати список у міру отримання;
- використовує $O(1)$ тимчасової пам'яті включаючи стек;
- мінусом є висока складність алгоритму - $O(n^2)$.

Схема програми сортування вставками представлена на рис. 3.

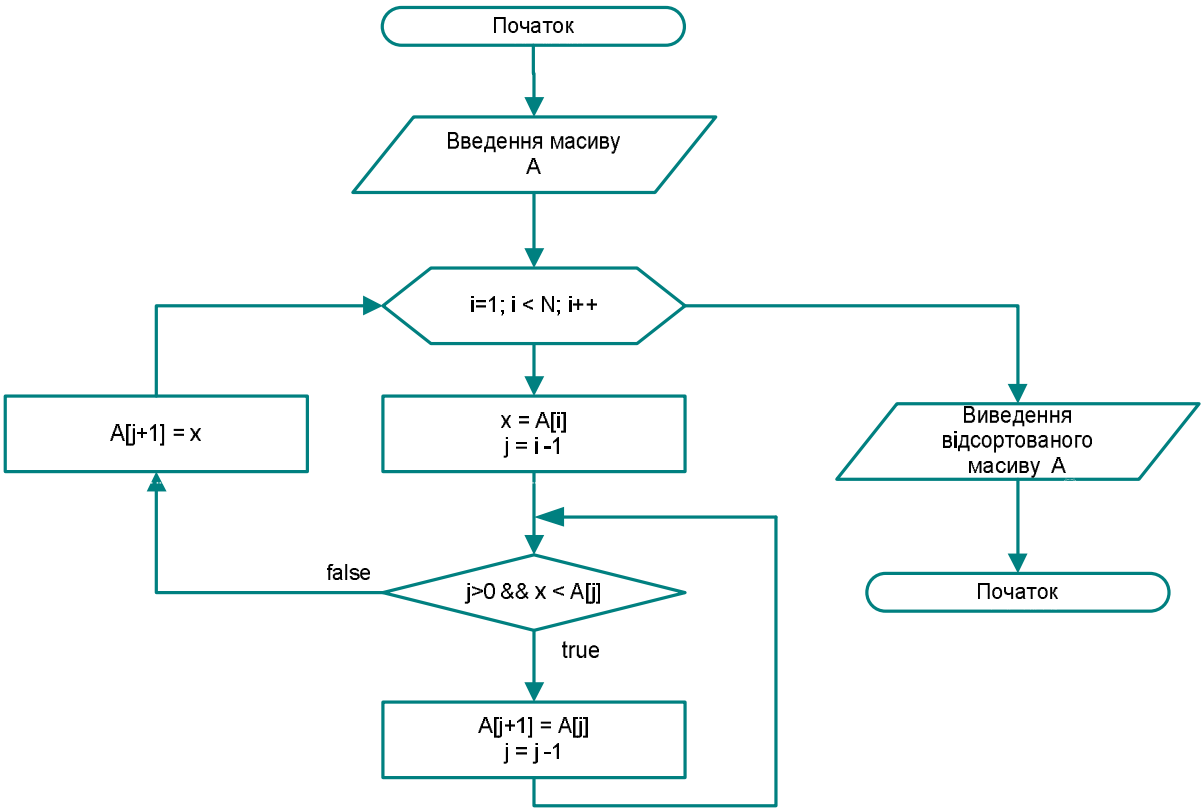


Рисунок 3 – Схема програми сортування вставками

Лінійний, або *послідовний пошук* – алгоритм знаходження заданого значення у списку. Пошук здійснюється простим порівнянням заданого значення зі значенням чергового елементу, що розглядається (як правило пошук здійснюється від першого елементу списку до останнього), і якщо значення співпадають пошук вважається закінченим.

Бінарний пошук – класичний алгоритм пошуку елемента у відсортованому масиві.

Нехай дано масив *A*, змінні *Lb* і *Rb* містять, відповідно, ліву і праву межі масиву(індекси), в якому знаходиться потрібний нам елемент, у змінній *key* зберігається шукане число. Робота алгоритму починається з дослідження середнього елемента масиву, індекс якого обчислюється як M=(Lb+Rb)/2. Якщо шукане значення *key* менше середнього елемента *A[M]*, здійснюється перехід до пошуку у правій половині масиву, де всі елементи менші щойно перевіреного. Іншими словами, значенням *Lb* стає *(M -1)* і на наступній ітерації здійснюється пошук лише у правій половині масиву. Таким чином, в результаті кожної перевірки вдвічі звужується область пошуку.

Схема програми бінарного пошуку представлена на рис. 4.

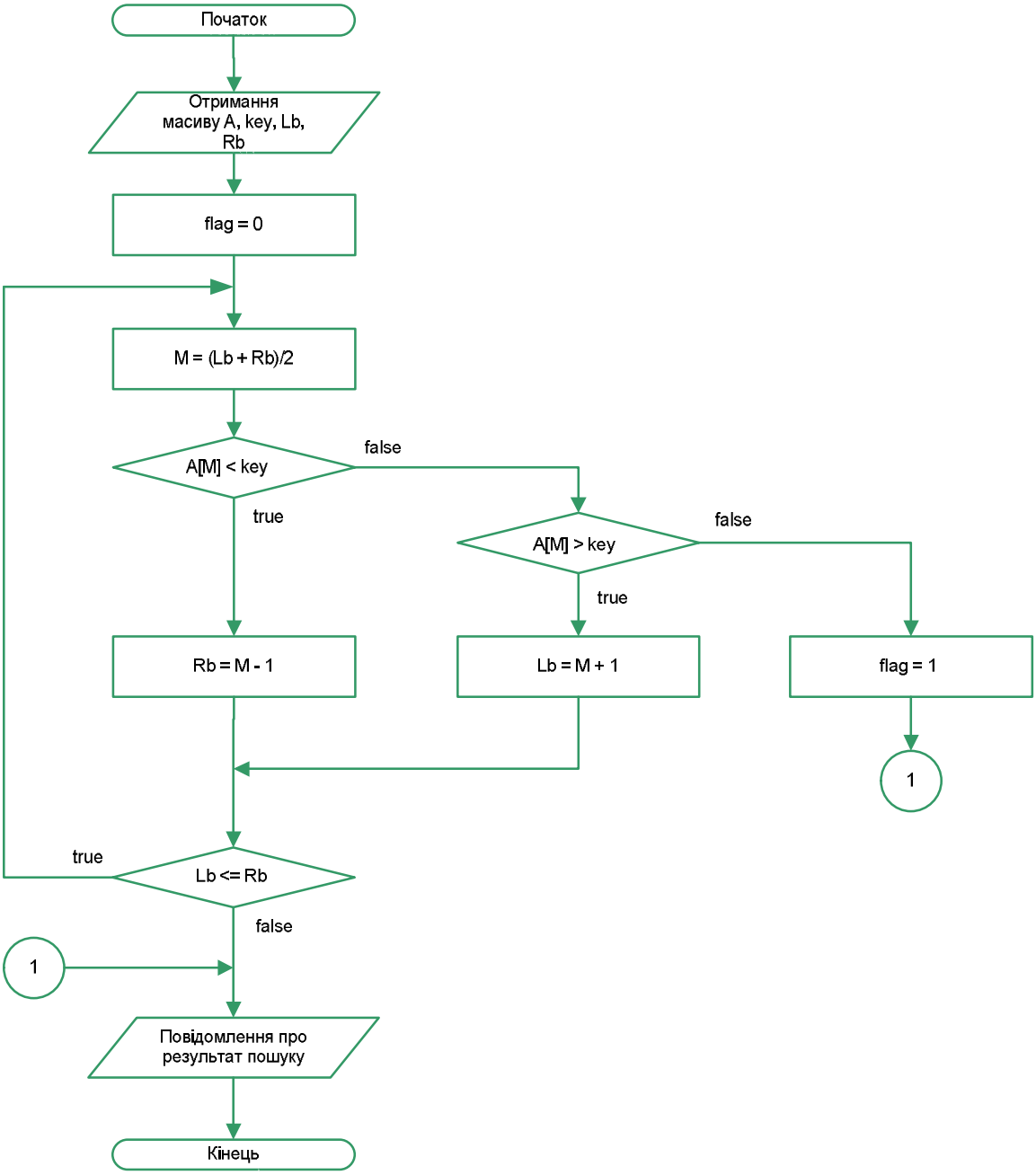


Рисунок 4 – Схема програми бінарного пошуку

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
   a. оголосити одновимірний статичний масив з десяти елементів типу int;
   b. заповнити масив з клавіатури;
   c. відсортувати масив та знайти введене з клавіатури число вказаними у варіантах алгоритмами;

d. вивести на екран відсортований масив;
e. вивести на екран кількість знайдених у масиві чисел, що відповідають умові пошуку та комірки, в яких ці числа були знайдені (для лінійного пошуку). У випадку відсутності чисел, які відповідають умові пошуку вивести відповідне повідомлення;
f. вивести на екран повідомлення про наявність чи відсутність чисел, що відповідають умові пошуку (для бінарного пошуку) .

2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. Зробити висновки.
6. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 1.
Відсортувати масив по зростанню. Алгоритм сортування – «бульбашковий». Метод пошуку – бінарний.

Варіант 2.
Відсортувати масив по спаданню. Алгоритм сортування – «бульбашковий». Метод пошуку – бінарний.

Варіант 3.
Відсортувати масив по зростанню. Алгоритм сортування – «вставкою». Метод пошуку – лінійний.

Варіант 4.
Відсортувати масив по спаданню. Алгоритм сортування – «вибором». Метод пошуку – лінійний.

Варіант 5.
Відсортувати масив по зростанню. Алгоритм сортування – «бульбашковий». Метод пошуку – лінійний.

Варіант 6.
Відсортувати масив по спаданню. Алгоритм сортування – «вставкою». Метод пошуку – бінарний.

Варіант 7.
Відсортувати масив по зростанню. Алгоритм сортування – «вибором». Метод пошуку – бінарний.

Варіант 8.
Відсортувати масив по спаданню. Алгоритм сортування – «вибором». Метод пошуку – бінарний.

Варіант 9.
Відсортувати масив по спаданню. Алгоритм сортування – «бульбашковий».

Підсумок

Після виконання лабораторної роботи студент повинен вміти створювати програми, які реалізують алгоритми сортування (бульбашковий, вибором, вставкою) та пошуку (лінійний, бінарний) на мові програмування С.

Контрольні питання

1. Що таке сортування масивів?
2. Які види сортування Ви знаєте?
3. Що таке зовнішнє сортування?
4. Що таке внутрішнє сортування?
5. Що таке бульбашкове сортування?
6. Використання сортування за зменшенням і алгоритми їх розв’язку?
7. Використання сортування за збільшенням і алгоритми їх розв’язку?
8. Що таке сортування методом вибору?
9. Що таке лінійний пошук?
10. Що таке бінарний пошук?

Контрольні вправи

1. Написати програму, яка сортує за зменшенням.
2. Написати програму, яка сортує за збільшенням.

Лабораторна робота №7. Багатовимірні масиви

Мета і задачі:
Навчитися створювати та відлагоджувати програми, в яких використовуються багатовимірні масиви на мові програмування С.

Теоретичні відомості і методичні вказівки

Багатовимірним називається масив, в якому кожен елемент адресується за допомогою декількох індексів. Окремим випадком багатовимірного масиву є двохвимірний масив – матриця. Використання декількох індексів впроваджується виключно для зручності і має суто логічну природу: насправді масиви будь-якої розмірності зберігаються в оперативній пам'яті ПК у вигляді одновимірної послідовності (рисунок 1).

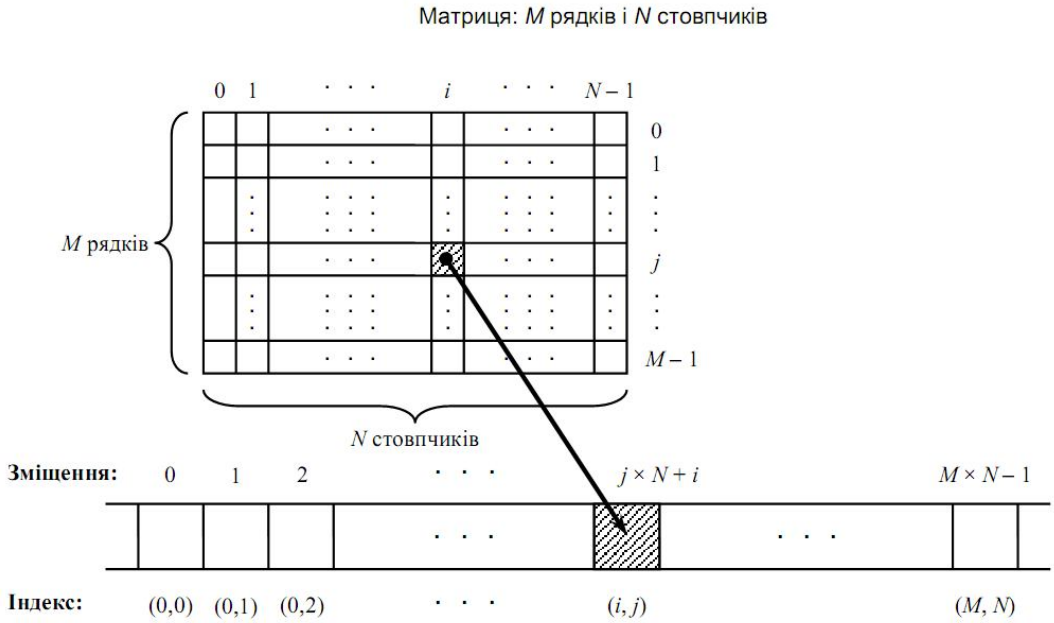


Рисунок 1 – Ілюстрація принципу зберігання двохвимірного масиву в пам'яті ПК

Оголошення та ініціалізація масивів
Оголошення масивів у загальному вигляді здійснюється за наступним форматом:

<тип даних> <назва> [розмірність 1][розмірність 2]... [розмірність N];

Приклади:
```
/* матриця чисел типу double (5 рядків і 10 стовпчиків)*/
double MatrixA[5][10];
/* трьохвимірний масив чисел типу char (7 на 7 на 10; всього 490 чисел) */
```

```
char Cube[7][7][10];
```

Ініціалізація масиву при оголошенні може бути виконана за допомогою так званого списку ініціалізаторів, наприклад:

```
float C[3][2] = {{0.1, 0.2}, {0.7, 1.3}, {1.9, -0.4}};
```

Обробка масивів

Зазвичай обробка масивів здійснюється у циклах; при цьому верхня і/або нижня границя циклу певним чином "прив'язується" до кількості елементів у масиві. Розглянемо, наприклад, фрагмент програми, в якому обчислюється сума елементів квадратної матриці x розміром 10×10, розташованих над головною діагоналлю:

```
int x[10][10];
...
// якщо розмір матриці зміниться, в наступному циклі
// необхідно буде змінити верхню границю
for (i = 0; i < 9; i++)
{
  // якщо розмір матриці зміниться, в наступному циклі
  // необхідно буде змінити верхню границю
  for (j = i + 1; j < 10; j++)
  {
    S += x[i][j];
  }
}
...
```

В даному прикладі зовнішній цикл змінює номер рядка – i – від найпершого до передостаннього (в останньому рядку над головною діагоналлю не розташовано жодного елементу), а внутрішній цикл змінює номер стовпчика – j – від (i + 1)-го до останнього. Якщо розмір матриці при розв'язанні конкретної задачі стане іншим, зміни слід буде внести в кожен цикл, яких в програмі може бути багато (як мінімум два цикли для заповнення матриці, два цикли – для виводу на дисплей), що є незручно. Якщо ж розмір матриці буде заданий проіменованою константою, достатньо буде змінити значення лише цієї константи:

```
...
const int n = 20; // зміни вносяться лише тут
int x[n][n];
...
// параметри наступного циклу
```

```
// не залежать від розміру матриці
for (i = 0; i < n - 1; i++)
{
 // параметри наступного циклу
 // не залежать від розміру матриці
 for (j = i + 1; j < n; j++)
 {
  S += x[i][j];
 }
}
...
```

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
2. оголосити двовимірний масив array[5][5] типу int;
3. заповнити масив з клавіатури;
4. вивести результати обчислень на екран.
5. Відкомпілювати та відлагодити програму.
6. Розробити набір тестів і перевірити роботу програми на них.
7. Зобразити схему програми.
8. Відповісти на контрольні запитання.
9. Зробити висновки.
10. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, схеми програми, висновків по роботі.

Варіанти завдань

Варіант 0.
Написати програму яка має виконувати наступні дії:
1. Підносити до квадрату всі від'ємні елементи масиву і добувати корінь з усіх додатних елементів.
2. Обчислювати суму зворотної діагоналі масиву.

Варіант 1.
Написати програму яка має виконувати наступні дії:
1. Обчислювати суму додатних та від'ємних елементів масиву.
2. Замінювати нулями від'ємні елементи масиву.

Варіант 2.
Написати програму яка має виконувати наступні дії:
1. Рахувати кількість елементів масиву, які знаходяться в заданому з клавіатури діапазоні. Якщо таких елементів більше половини замінити їх одиницями.

2. Обчислювати кількість парних чисел масиву.

Варіант 3.
Написати програму яка має виконувати наступні дії:
1. Здійснювати сортування елементів головної діагоналі масиву по зростанню.
2. Обчислювати кількість непарних чисел масиву.

Варіант 4.
Написати програму яка має виконувати наступні дії:
1. Всі парні елементи масиву, які знаходяться нижче головної діагоналі замінювати заданим з клавіатури числом.
2. Всі непарні елементи масиву замінювати нулями.

Варіант 5.
Написати програму яка має виконувати наступні дії:
1. Обчислювати суму елементів, які знаходяться вище побічної діагоналі масиву.
2. Елементи які знаходяться нижче побічної діагоналі масиву заповнювати нулями.

Варіант 6.
Написати програму яка має виконувати наступні дії:
1. Обчислювати добуток елементів масиву, що знаходяться в заданому діапазоні для кожного стовпця масиву.
2. Побічну діагональ масиву заповнювати нулями.

Варіант 7.
Написати програму яка має виконувати наступні дії:
1. Обчислювати різницю між сумами елементів масиву, які знаходяться вище і нижче головної діагоналі масиву.
2. Головну діагональ масиву заповнювати нулями.

Варіант 8.
Написати програму яка має виконувати наступні дії:
1. Всі від'ємні елементи масиву замінювати заданим з клавіатури числом.
2. Обчислювати суму кожної стрічки масиву.

Варіант 9.
Написати програму яка має виконувати наступні дії:
1. Міняти місцями дві стрічки, в яких знаходяться мінімальний і максимальні елементи масиву. Якщо мінімальний і максимальний

елементи знаходяться в одній стрічці виводити на екран відповідне повідомлення.
2. Обчислювати середнє арифметичне елементів масиву.

Підсумок

Після виконання лабораторної роботи студент повинен вміти створювати програми, в яких використовуються багатовимірні масиви на мові програмування С.

Контрольні питання

1. Що таке багатомірний масив ?
2. Для чого використовуються багатомірні масиви ?
3. Скільки індексів може мати багатомірний масив ?
4. Як ініціалізується багатовимірний масив ?
5. Як ініціалізувати двовимірний масив при об'явленні ?

Контрольні вправи

1. Заповнити двовимірний масив розміром 5 на 5 одиницями в шаховому порядку?
2. Заповнити двовимірний масив розміром 5 на 5 одиницями по головній діагоналі?
3. Заповнити двовимірний масив розміром 5 на 5 одиницями по побічній діагоналі?

Лабораторна робота №8. Функції

Мета і задачі:
Навчитися створювати та відлагоджувати програми, в яких використовуються функції на мові програмування С.
Теоретичні відомості і методичні вказівки
Функція - це група операторів, що виконується, коли він викликається в якийсь момент програми. Формат функції наступний:
<ім'я типу> <назва функції>(параметр1, параметр2, ...) {тіло функції}

Приклад програми з функцією, яка здійснює додавання двох чисел:

```
#include <stdio.h>

int addition (int a, int b)
{
 int r;
 r=a+b;
 return (r);
}
int main()
{
 int z;
 z = addition(5,3);
 printf("Функція повертає значення %i", z);
 return 0;
}
```

Функція *main* починається з оголошення змінної *z* типу *int*. Відразу після цього здійснюється виклик функції **addition**. Схожість між структурою виклику функції та оголошенням самої функції зображена на рис. 1:

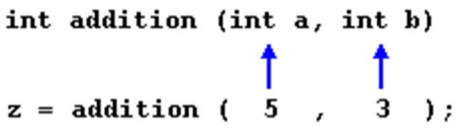


Рис 1.

У функції main здійснюється виклик функції *addition*, їй передаються через параметри два значення: 5 і 3, які відповідають параметрам *int a i int b* в оголошенні функції *addition*.
В точці, в якій функція *addition* викликається з *main* управління передається функції *addition*. Значення обох аргументів (5 і 3) будуть скопійовані в локальні змінні *int a* і *int b* в межах функції *addition*.

Функція *addition* оголошує локальну змінні *int r,* виконує вираз *r=a+b*, в результаті чого у r буде присвоєно значення 8, так як значення параметрів *a i b*, були 5 і 3 відповідно.
Наступний рядок коду:
return (r);
завершує функцію *addition*, і повертає значення назад у функцію main. Далі програма виконується з того ж місця, на якому вона була перервана викликом функції *addition*. Значення що повертається з функції є значенням функції, що викликається (рис 2.). Щоб пояснити це інакше, можна собі уявити, що виклик функції addition (5,3) буквально замінюється значенням я, яке вона повертає (8).

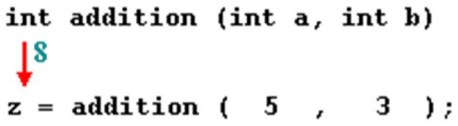


Рис. 2

Наступний рядок коду виводить на екран значення змінної z:
printf("Функція повертає значення %i", z);
Порядок виконання і звітування
1. Написати програму, яка демонструє роботу функції заданої відповідно до варіанту використавши середовище програмування Dev-C++ 4.0.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. Зробити висновки.
6. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 0.
Написати рекурсивну функцію **power (base, exponent)**, яка повертає значення $base^{exponent}$.
Наприклад, power(3, 4) = 3 * 3 * 3 * 3 . Нехай *exponent* є цілим, більшим або рівним 1.
Підказка: крок рекурсії міг би використати співвідношення $base^{exponent} = base * base^{exponent}$, а завершальною умовою буде випадок, коли значення *exponent* стане рівним 1.

Варіант 1.
Як відомо, властивість послідовності чисел Фібоначі: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... полягає у тому, що кожне наступне число є сумою двох попередніх.

Написати нерекурсивну функцію **fibonacci (n)**, що обчислює n-не число Фібоначі.

Варіант 2.
Найбільший загальний дільник (НОД) двох цілих чисел є самим більшим цілим числом, на яке ділиться кожне із двох чисел.
Написати функцію **gcd,** яка повертає найбільший загальний дільник двох цілих чисел.

Варіант 3.
Найбільший загальний дільник цілих чисел *x* і *y* є найбільшим цілим числом, на яке діляться і *x* і *y*.
Написати рекурсивну функцію **gcd,** що повертає найбільший загальний дільник *x* і *y* та програму, яка використовує цю функцію. Рекурсивне визначення функції **gcd** виглядає в такий спосіб: якщо *y* дорівнює 0**,** то *gcd(x, y)* є *x*, інакше *gcd(x, y)* є *gcd(y, x % y)*, де % — операція ділення по модулю.

Варіант 4
Ціле число називають простим, якщо воно ділиться тільки на 1 і на саме себе. Наприклад, 2, 3, 5 і 7 є простими, а 4, 6, 8 і 9 - немає.
Написати функцію, яка визначає, чи є число простим. Використайте цю функцію в програмі, що знаходить і друкує всі прості числа в діапазоні від 1 до 10000.

Варіант 5.
Ціле число називають досконалим числом, якщо сума його дільників, включаючи 1 (але не саме число), дорівнює цьому числу. Наприклад, 6 є досконалим числом, оскільки 6 = 1+ 2 + 3.
Написати функцію **perfect,** яка визначає, чи є її параметр досконалим числом. Використайте цю функцію в програмі, що знаходить і друкує всі досконалі числа в діапазоні від 1 до 1000. Надрукуйте всі дільники для кожного досконалого числа, щоб переконатися, що число дійсно є досконалим.

Варіант 6.
Написати функцію, яка одержує час у якості трьох цілих аргументів (години, хвилини й секунди) і повертає число секунд із моменту, коли годинник «пробив 12».

Варіант 7.

Написати функцію, яка одержує ціле значення і повертає число із оберненим порядком цифр. Наприклад, для числа 7631 функція повинна повернути значення 1367.

Варіант 8.
Написати функцію **gualityPoints**, яка вводить середній бал студента й повертає 4, якщо середній бал студента становить 90-100, 3, якщо 80-89, 2, якщо 70-79, 1, якщо 60-69, і 0, якщо середній бал нижче ніж 60.

Варіант 9.
Написати програму, яка моделює підкидання монети. Для кожного підкидання монети програма повинна друкувати слова «Heads» або «Tails» (орел або решка). Нехай програма підкине монету 100 разів і порахує число випадань для кожної сторони монети. Програма повинна викликати окрему функцію **flip,** що не одержує ніяких аргументів і повертає 0 для решки й 1 для орла.

Підсумок
Після виконання лабораторної роботи студент повинен вміти створювати програми, в яких використовуються функції на мові програмування С.
Контрольні питання

1. Що таке функція ?
2. Де і як визиваються функції ?
3. Що є аргументом функції ?
4. Написати загальний формат знаходження функції ?
5. Що може зробити визвана функція ?
6. Що таке прототип функції ?
7. Для чого призначені компілятори в функції ?
8. Де знаходяться прототипи функцій **rand** і **srand** , і що це таке ?

Контрольні вправи

1. Знайдіть помилку у фрагменті програми:

a. int g (void)
```
{
  printf (“Inside function g\n”);
   {
     printf(“Inside function h\n”);
   }
}
```
b. int sum (int x, int y)
```
{
```

```
    int result;
    result = x + y;

      c. void f ( float a ) ;
{
    float a ;
    printf ( “%f”, a ) ;   }
```

2. Напишіть функцію distance, яка обчислює відстань між двома точками з координатами (x1, y1) і (x2, y2). Всі числа і значення що повертаються повинні мати тип **float**.
3. Написати функцію, яка обчислює та повертає максимальне значення з двох переданих через параметри.
4. Написати функцію, яка обчислює та повертає мінімальне значення з двох переданих через параметри.
5. Підрахувати в одномірному масиві цілого типу розміром 100 елементів кількість нульових значень.
6. Написати функцію з двома параметрами логічного типу, який повертає значення у відповідності по наступній таблиці істинності (таблиця 1).

Таблиця 1

Параметри		Повернений результат
Перший	Другий	
False	False	False
False	True	False
True	False	True
True	True	False

Параметри і результати - цілого типу: false відповідає нулю і true - нульовим значенням.

Лабораторна робота №10. Покажчики

Мета і задачі:

Навчитися створювати та відлагоджувати програми, в яких використовуються покажчики на мові програмування С.

Теоретичні відомості і методичні вказівки

Покажчик - це змінна, значенням якої є адреса іншої змінної. Так як покажчик може посилатися на змінні різних типів, з покажчиком у мові Сі зв'язується тип того об'єкта, на який він посилається. Для опису покажчиків використовується операція непрямої адресації *. Наприклад, покажчик цілого типу *uk* описується так:

```
int * uk;
```

Унарна операція &, застосована до деякої змінної, показує, що нам потрібен адрес цієї змінної, а не її поточне значення. Якщо змінна *uk* оголошена як покажчик, то оператор присвоювання *uk = & x* означає: "взяти адресу змінної *x* і привласнити його значення змінній-вказівником *uk*".

Унарна операція * застосована до покажчика, забезпечує доступ до вмісту комірки пам'яті, на яку посилається покажчик. Наприклад, * *uk* можна описати словами як "те, що міститься за адресою, на який вказує *uk*". Покажчики можуть використовуватися в виразах. Якщо. наприклад, змінна *uk* вказує на ціле *x*, то * *uk* може у всіх випадках використовуватися замість *x*; так, * *uk +1* збільшує *x* на одиницю, а * *uk = 0* рівносильне *x = 0*. Два оператора присвоювання *uk = &x; y =* uk;* виконує те ж саме, що і один оператор y = x. Користь від застосування покажчиків в таких ситуаціях, м'яко кажучи, невелика.

Найбільш повно їх переваги при обробці масивів і, зокрема, символьних рядків. Покажчики та масиви тісно пов'язані один з одним. Перш ніж розглянути цей зв'язок детально, зазначимо, що якщо *uk* - деякий покажчик, то *uk* ++ збільшує його значення і він тепер вказує на наступний, сусідній об'єкт. Значення *uk* використовується у виразі, а потім збільшується. Аналогічно визначаються операції *uk--*, *++uk*, *--uk*. У загальному випадку покажчик *uk* можна додавати до цілого числа *i*. Оператор *uk += i* пересуває посилання на *i* елементів щодо поточного значення. Ці конструкції підпорядковуються правилам адресної арифметики.

А тепер повернемося до масивів. Нехай є опис *int a[5]*. Воно визначає масив розміром 5 елементів, тобто п'ять послідовних розташованих комірок пам'яті *a[0]*, *a[1]*, *a[2]*, *a[3]*, *a[4]*. Адреса і-го елемента масиву дорівнює сумі адреси початкового елемента масиву і зміщення цього елемента на *i* одиниць від початку масиву. Це досягається індексуванням: *a[i]* - і-й елемент масиву. Але доступ до будь-якого елементу масиву може бути виконаний і за допомогою покажчиків, причому, більш ефективно. Якщо *uk* - покажчик на ціле, описаний як *int * uk*, то *uk* після виконання

операції *uk = & a[0]* містить адресу *a [0]*, а *uk + i* вказує на і-й елемент масиву. Таким чином, *uk + i* є адресою *a[i]* комірки масиву, а * *(uk + i)* - вмістом і-го елемента (операції * і & є більш пріоритетними, ніж арифметичні операції). Оскільки ім'я масиву в програмі ототожнюється з адресою його першого елемента, то вираз *uk = & a [0]* еквівалентно такому: *uk = a*. Тому значення *a[i]* можна записати як * *(a + i)*. Застосувавши до цих двох елементів операцію взяття адреси, отримаємо, що *& a[i]* і *a + i* ідентичні.

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
    a. для рішення задачі використати динамічний масив;
    b. масив задати з клавіатури;
    c. для рішення задачі написати функцію, яка буде оперувати елементами масиву. Параметри в функцію передавати через покажчик, крім окремо оговорених в варіантах завдань випадків.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. Зробити висновки.
6. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 0.
Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися всі елементи, модуль яких не перевищує 1, а потім всі інші.

Варіант 1.
Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися нульові елементи, а потім всі інші.

Варіант 2.
Написати програму, яка видаляє всі елементи масиву, модуль яких не перевищує 1; елементи, які звільнилися в кінці масиву заповнюються нулями.

Варіант 3.
Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися елементи, які відрізняються від максимального не більш ніж на 20%, а потім усі інші.

Варіант 4.
Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися додатні числа, а потім від'ємні.

Варіант 5.
Написати програму, яка буде упорядковувати елементи масиву за зростанням модулів елементів.

Варіант 6.
Написати програму, яка буде змінювати порядок елементів в масиві на зворотній.

Варіант 7.
Написати програму, яка буде перетворювати масив таким чином, щоб нульові елементи розташовувалися після всіх інших.

Варіант 8.
Написати програму, яка видаляє всі елементи, модуль яких знаходиться в інтервалі [a,b]; елементи які вивільнилися в кінці масиву заповнити нулями.

Варіант 9.
Написати програму, яка перетворює масив таким чином, щоб в першій його половині знаходилися елементи, які знаходилися в непарних позиціях, а в другій половині – елементи які знаходилися в парних позиціях.

Підсумок
Після виконання лабораторної роботи студент повинен вміти створювати програми, в яких для обчислювальних процесів використовуються покажчики на мові програмування С.
Контрольні питання
1. Що таке покажчик ?
2. Назвіть три основні значення покажчика ?
3. Що повинно бути операндом операції ?
4. Що таке операція * ?
5. Назвіть чотири способи передачі покажчика в функцію ?
6. Які операції можуть виконуватися разом з покажчиками ?
7. Для чого призначений покажчик типу **void *** ?
8. Що таке ім'я масиву ?

Контрольні вправи

1) Що надрукує наступна програма?

```
# include <stdio.h>
    int    Array [ ] = { 0, 4, 5, 2, 3 };
    int main  ( void)
    {
        int    Index, *Pointer;
        for ( Index = 0; Index <= 4; Index +=2 )
              printf ( " %3d", * (Array+Index--) );
        printf ( "\n" );
        Pointer = Array + 1;
        for ( Index = 0; Index <= 2; )
              printf ( " %3d", Pointer [ ++Index ] );
        printf ( "\n" );
        return 0;
    }
```

2) Що надрукує наступна програма?

```
    # include <stdio.h>
    int    Array [ ] = { 1, 2, -7, 4, 3  };
    int main  ( void)
    {
        int    Index, *Pointer;
        for ( Index = 0; Index <= 4; Index +=2 )
             printf ( " %3d", Array[ Index] );
        printf ( "\n" );
        Pointer = Array + 1;
        for ( Index = 0; Index <= 2;  ++Index )
             printf ( " %3d", Pointer [ ++Index ] );
        printf ( "\n" );
        return 0;
    }
```

3) Що надрукує наступна програма?

```
# include <stdio.h>
    int    Array [ ] = { 1, 4, 7, 2, 3  };
    int main  ( void)
    {
        int    Index, *Pointer;
        for ( Index = 1; Index <= 4; Index +=1 )
             printf ( " %3d", Array[ ++Index] );
```

```
    printf ( "\n" );
    Pointer = Array ;
    for ( Index = 0; Index <= 2;  ++Index )
        printf ( " %3d", Pointer [ Index++ ] );
    printf ( "\n" );
    return 0;
}
```

4) Що надрукує наступна програма?

```
# include <stdio.h>
    int    Array [ ] = { 1, 4, 5, 12, 3};
    int main  ( void)
    {
        int    Index, *Pointer;
        for ( Index = 1; Index <= 4; Index +=1 )
            printf ( " %3d", * (Array+Index++) );
        printf ( "\n" );
        Pointer = Array + 1;
        for ( Index = 0; Index <= 2;  ++Index )
            printf ( " %3d", Pointer [ ++Index ] );
        printf ( "\n" );
        return 0;
    }
```

5) Що надрукує наступна програма?

```
  # include <stdio.h>
    int    Array [ ] = { 0, 4, 5, 2, 3  };
    int main  (void)
    {
        int    Index, *Pointer;
        for ( Index = 0; Index <= 4; Index +=2 )
            printf ( " %3d", * (Array+Index++) );
        printf ( "\n" );
        Pointer = Array + 1;
        for ( Index = 0; Index <= 3;  Index++ )
            printf ( " %3d", Pointer [ ++Index ] );
        printf ( "\n" );
        return 0;
    }
```

Лабораторна робота №11. Робота з файлами

Мета і задачі:

Навчитися створювати та відлагоджувати програми, у яких здійснюється робота з файлами (зчитування/запис) на мові програмування С.

Теоретичні відомості і методичні вказівки

Файл – це впорядкована послідовність однотипних компонент, розташованих на зовнішньому носії. Файли призначені для зберігання інформації, а обробка цієї інформації здійснюється програмами.

Використання файлів доцільно у випадку:

- довготривалого зберігання даних
- доступу різних програм до одних і тих же даних
- обробки великих масивів даних, які неможливо повністю розмістити в оперативній пам'яті комп'ютера

У мові С використовуються такі функції бібліотеки stdio.h для роботи з текстовими файлами: **fopen**, **fscanf**, **fprintf**, **fclose**.

Для роботи з текстовими файлами спочатку потрібно оголосити файлову змінну, яка є покажчиком на файл :

FILE * <файлова змінна>;

Наприклад, оголошення файлової змінної f :

```
FILE *f;
```

Потім потрібно відкрити файл і зв'язати його з файловою змінною використавши функцію **fopen**:

файлова змінна = fopen ("шлях до файлу", «режим доступу»);

Наприклад, відкриваємо файл in.txt для зчитування:

```
f = fopen("D:\\in.txt", "r");
```

Функція **fopen** повертає покажчик на структуру типу FILE при успішному відкритті файлу, і NULL в протилежному випадку

В таблиці 1 вказані режими доступу, які застосовуються у функції fopen.

Таблиця 1

Режим доступу	Опис
r	Файл відкривається для читання
r+	Файл відкривається для читання і запису
w	Відкривається пустий файл для запису
w+	Відкривається пустий файл для читання та запису
a	Файл відкривається для дозапису в кінець файлу

a+	Файл відкривається для читання та дозапису в кінець файлу

Режим відкриття може також містити символи t (текстовий файл) и b (двійковий файл), вказуючи на тип файлу, який відкривається: rb, wb, ab, rt, at, rb+, wb+, ab+ тощо.
Для перевірки існування чи правильності відкриття файлу можна використати такий фрагмент коду:

```
if ((fopen("D:\\in.txt", "r"))==NULL)

{printf("Помилка при відкритті файлу");

return 0;

}
```

Приклад відкриття файлу out.txt для запису:

```
f2 = fopen("D:\\out.txt", "w");
```

Загальний прототип функції зчитування з файлу **fscanf**:

fscanf(файлова змінна, стрічка форматування, список адрес змінних);

Приклад, зчитування з файлу даних у змінні *a* і *b* типу int:

```
fscanf(f,"%i %i",&a, &b);
```

Загальний прототип функції запису у файл **fprintf**:

fprintf(файлова змінна, стрічка форматування, список змінних);

Приклад, запису у файл значень змінних *a* і *b* типу int:

```
fscanf(f2,"%i %i",a, b);
```

Загальний прототип функції закриття файлу **fclose**:

fclose(файлова змінна);

Приклад, закриття файлу:

```
flose();
```

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
2. для збереження даних про планшетні сканери описати структуру наступного вигляду:

```
i.   struct scan_info{
ii.  char model [25];     //найменування моделі
iii. int price;          // ціна
iv.  double x_size;      // горизонтальний розмір області сканування
v.   double y_size;      // вертикальний розмір області сканування
vi.  int optr;           // оптичний дозвіл
vii. int grey;           // число градацій сірого
viii. };
```

3. структура файлу: спочатку у файлі розміщається значення типу int, що визначає кількість зроблених у файлі записів; далі в кожній новій стрічці розміщуються записи про сканери.
4. Відкомпілювати та відлагодити програму.
5. Розробити набір тестів і перевірити роботу програми на них.
6. Відповісти на контрольні запитання.
7. Зробити висновки.
8. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 1.

1. Написати функцію, що записує у текстовий файл дані про сканер із приведеної структури.
2. Написати функцію, що витягає з цього файлу дані про сканер у структуру типу scan_info. Обов'язковий параметр - номер необхідного запису. Функція повинна повертати нульове значення, якщо читання пройшло успішно, і -1 у протилежному випадку.
3. Привести приклад програми, що створює файл із даними про сканери, (дані вводяться з клавіатури) - 6-8 записів, і виводить на дисплей дані про запитуваний запис.

4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускається.

Варіант 2.

1. Написати функцію, що записує в текстовий файл дані про сканер з приведеної структури.
2. Написати функцію, що сортує записи в описаному вище текстовому файлі по одній з наступних характеристик: ціна або число градацій сірого. Обов'язковий параметр - ознака, що задає критерій сортування.
3. Привести приклад програми, що створює файл із даними про сканери (дані вводяться з клавіатури) не менш восьми записів і здійснюючи його сортування.
4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускаєтеся.

Варіант 3.

1. Написати функцію, що сортує записи в описаному вище текстовому файлі по найменуванню моделі сканера.
2. Привести приклад програми, що створює файл із даними про сканери, (дані вводяться з клавіатури) не менш восьми записів, і здійснює його сортування.
3. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускається.

Варіант 4 .

1. Написати функцію, що динамічно виділяє пам'ять під масив структур (не менше шести елементів), заповнює його даними в режимі діалогу і записує масив у текстовий файл.
2. Написати функцію, що витягає дані про сканер з описаного вище текстового файлу в структуру типу scan info. Обов'язковий параметр - номер необхідного запису. Функція повинна повертати нульове значення, якщо зчитування пройшло успішно, і -1 в іншому випадку.
3. Привести приклад програми, що створює файл із даними про сканери , (дані вводяться з клавіатури), не менш восьми записів, і здійснює вивід на дисплей даних про необхідний запис.
4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускається.

Варіант 5.

Запис може здійснюватися в будь яку позицію, при чому, якщо між записом, що вводиться, і останнім (чи початком файлу) існують пропуски, вони заповнюються нулями.

Написати функцію, що "ущільнює" описаний вище текстовий файл шляхом видалення з нього записів, що містять усі нулі.
Привести приклад програми, що створює файл із даними про сканери (дані вводяться з клавіатури) не менше шести записів, і здійснює його ущільнення.
Усі необхідні дані для функцій повинні передаватися їм у якості параметрів. Використання глобальних змінних у функціях не допускається.

Варіанті 6.
1. Написати функцію, що динамічно виділяє пам'ять під масив структур (не менше шести елементів), заповнює його даними в режимі діалогу і записує масив у тестовий файл.
2. Написати функцію, що запитує дані про сканер у режимі діалогу і заміщає запис у текстовому файлі по заданому номеру. Обов'язковий параметр - номер запису, що заміщається. Функція повинна повертати нульове значення, якщо запис пройшов успішно, і -1 у іншому випадку.
3. Привести приклад програми, що створює файл із даними про сканери (дані вводяться з клавіатури) не менш восьми записів, і здійснює вставку нових даних про сканер.
4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускається.

Варіант 7.
1. Написати функцію, що записує в текстовий файл дані про сканер із приведеної структури.
2. Написати функцію, що вводить дані про сканер із клавіатури в структуру типу scan_info, і якщо дані про цей сканер відсутні у файлі, розміщує вміст структури в кінець файлу; у протилежному випадку, видає відповідне повідомлення.
3. Привести приклад програми, що створює файл із даними про сканери (дані вводяться з текстового файлу) - 6-8 записів, і доповнює файл записами про 2-3 сканери, що вводяться з клавіатури.
4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускається.

Варіант 8.
1. Написати функцію, що записує в текстовий файл дані про сканер з приведеної структури.
2. Написати функцію, що вводить дані про сканер із клавіатури в структуру типу scan_info і дописує у кінець файлу ці дані.
3. Привести приклад програми, що створює файл із даними про сканери (дані вводяться з текстового файлу) - 6-8 записів, і доповнює цей файл 1-2 новими записами, що вводяться з клавіатури.

4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних у функціях не допускається.

Варіант 9.

1. Написати функцію, що запитує кількість сканерів, інформація про які буде вводитися, динамічно виділяє пам'ять під масив структур відповідного розміру і заповнює його даними в режимі діалогу (з клавіатури).
2. Написати функцію, що записує даний масив у створюваний текстовий файл. Якщо ціна сканера менше 200, то дані про цей сканер у файл не записуються. Інформація про інші сканери міститься в бінарному файлі, при чому спочатку пишуться дані про всі сканери, ім'я яких починається з заголовної букви, а потім – із прописної.
3. Привести приклад програми, що створює файл із даними про сканери і здійснює вивід на дисплей даних про необхідний запис (або всіх, або по номеру).
4. Усі необхідні дані для функцій повинні передаватися їм як параметри. Використання глобальних змінних в функціях не допускається.

Підсумок

Після виконання лабораторної роботи студент повинен навчитися створювати програми, у яких здійснюється робота з файлами (зчитування/запис) на мові програмування С.

Контрольні питання

1. Що таке файл ?
2. Для чого призначений файл і його використання ?
3. Що таке символи ?
4. Що таке поле ?
5. Що називається найбільш популярний тип організації записів в файлі ?
6. Що називають групою зв’язаних між собою файлів ?
7. Назвіть функції мови С які використовуються для роботи з файлами ?

Контрольні вправи

1. В файлі операційної системи *“Test1.in”* є 5 стрічок, кожна з яких має довжину сторін прямокутників (значення довжини розділені двома пробілами).

Написати:

- визначення масиву структур для зберігання вказаних довжин сторін прямокутника, їхню площу і периметр;
- фрагмент програми для читання довжини сторони прямокутника із файла операційної системи “*Test1*.in”;

- фрагмент програми, яка обчислює і друкує площі і периметри прямокутників у файл операційної системи *“Test1.out”*.

2. В текстовому файлі “*Test2*.in” є інформація про квартиру, яка має наступний вигляд:

```
01  Комната        15
      .....
05  Кухня          5
```

Кожна стрічка має відомості по одній кімнаті: перші дві позиції – порядковий номер кімнати, третя позиція – “пробіл”, з поз. 4 починається назва кімнати довжиною не більше 15 символів, з поз. 21 – метраж кімнати.
Написати:

- визначення масиву структур для зберігання вказаних даних і фрагмент програми для читання даних про квартирі із файлу *“Test2.in”*;
- фрагмент програми для знаходження і друку загального метражу даної квартири в файл *“Test2.out”*

3. Що буде виведено у файл наступним фрагментом програми:

```
float   r;
int     i = 17;
r = 1.5f * 2.0elf;
fprintf ( stdout, “*r=%5.2e^%s^*i=%-+d\n*%-3s\n”, r, “_”, i, “*”,);
```

4. Що буде виведено у файл наступним фрагментом програми:

```
float   r;
int     i = 17;
r = 1.5f * 2.0;
fprintf ( stdout, “*r=%5.2f^%5s^*i=%-+10d\n*%-30s\n”, r, “_”, i, “*”,);
```

Лабораторна робота №12. Обробка символьної інформації

Мета і задачі:

Навчитися створювати та відлагоджувати програми, у яких здійснюється обробка символьної інформації на мові програмування С.

Теоретичні відомості і методичні вказівки

Якщо в програмі треба працювати з рядками, їх оголошують як масиви символів. В такому масиві для кожного символу рядка (правильніше: коду символу) відводиться своя комірка, а останній елемент містить символ кінця рядка – „\0“ (код нуль).

Для зберігання кодів символів найбільш добре підходить тип даних, який в літературі називають символьним: **char**.

Символьний масив оголошується та ініціалізується за схемою, загальною для масивів довільного типу, наприклад:

```
/* оголошення символьного масиву на 50 елементів */
char FirstName[50];
// Масив Greeting1 містить коди символів ‘H’, ‘e’, ...*/
char Greeting1[] = {„H“, „e“, „l“, „l“, „o“};
/* Масив Greeting2 ідентичний за вмістом масиву Greeting1 тому що код літери ‘H’ – 72, ‘e’ – 101, ... */
char Greeting2[] = {72, 101, 108, 108, 111};
```

Для зручності компілятори C/C++ дозволяють ініціалізувати символьні масиви не лише звичайним шляхом, продемонстрованим в попередньому прикладі, але й наступним чином:

```
char StreetName[] = “вул. Метробудівська 5-а”;
```

Для масиву StreetName компілятор автоматично виділяє пам’ять у розмірі 24 байта (довжина ініціалізуючого рядка плюс один байт – символ завершення рядка).

При вводі рядка з клавіатури за допомогою функції scanf оператор отримання адреси – & – не потрібен, так як ідентифікатор масиву фактично являє собою адресу його початку:

```
char FirstName[20];
...
scanf (“%s”, FirstName);
```

Обробка символьних масивів – рядків в цілому виконується аналогічно обробці масивів іншого типу, однак комплект засобів розробки C/C++ включає заготовочний модуль **string.h**, до складу якого входять функції,

спеціалізовані для обробки рядків. Прототипи деяких, найбільш використовуваних функцій цього модуля представлені в таблиці 1.

Таблиця 1

Прототип функції	Опис
void *memcpy (void *dest, const void *src, int n);	Копіювання одного масиву в інший
char *strcpy (char *dest, const char *src)	Копіювання одного рядка в інший
char *strcat (char *dest, const char *src)	Конкатенація (зчеплення) двох рядків
char *strchr (const char *s, int c)	Пошук символу в рядку
int strcmp (const char *s1, const char *s2)	Порівняння рядків
int strlen (const char s)	Обчислення довжини рядка
char *strlwr (char *s)	Перетворення символів рядка у маленькі
char *strrev (char *s)	Інверсія рядка (перестановка всіх символів у зворотному порядку)
char *strset (char *s, int ch)	Заповнення рядка заданим символом
char strstr (const char *s1, const char *s2)	Пошук одного рядка в іншому
char *strupr (char *s)	Перетворення символів рядка у великі

Знак „*“ (зірка) в таблиці означає, що аргументом або результатом роботи функції є адреса рядка (так званий покажчик), з яким можна працювати як з ідентифікатором відповідного масиву символів.

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
2. за допомогою текстового редактора створити файл, що містить текст, довжина якого не перевищує 1000 символів (довжина рядка тексту не повинна перевищувати 70 символів);
3. ім'я файлу повинне мати розширення dat.
4. Відкомпілювати та відлагодити програму.
5. Розробити набір тестів і перевірити роботу програми на них.

6. Відповісти на контрольні запитання.
7. Зробити висновки.
8. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 0.
Написати програму, яка виводить на екран символ, який зустрічається у файлі максимальне число разів.

Варіант 1.
Написати програму, яка визначає скільки разів у файлі зустрічаються символ введений з клавіатури.

Варіант 2.
Написати програму, яка знаходить слова з файлу, які містять цифрові символи. Вивести знайдені слова на екран монітору.

Варіант 3.
Написати програму, яка визначає кількість слів у файлі, які закінчуються на словосполучення, що задається з клавіатури. Вивести знайдені слова на екран монітору.

Варіант 4.
Написати програму, яка визначає кількість букв, кожного слова з файлу. Вивести на екран слово мінімальної довжини.

Варіант 5.
Написати програму, яка виводить на екран першу та останню букву кожного слова з файлу.

Варіант 6.
Написати програму, яка визначає, скільки разів у тексті з файлу зустрічається кожен з символів.

Варіант 7.
Написати програму, яка визначає кількість букв, кожного слова з файлу. Вивести на екран слово максимальної довжини.

Варіант 8.
Написати програму, яка виводить на екран слова з файлу у стовпчик.

Варіант 9.
Написати програму, яка у файлі всі входження *a* замінює на *b*. Словосполучення *a* та *b* задаються з клавіатури.

Підсумок
Навчитися створювати програми, у яких здійснюється обробка символьної інформації на мові програмування С.
Контрольні питання
1. Що виконує функція islower?
2. Що виконує функція atoll?
3. Що виконує функція gets?
4. Що виконує функція sprintf?
5. Що виконує функція strstr?
6. Що виконує функція atoi?
7. Що виконує функція atof?
8. Що виконує функція putchar?
9. Що виконує функція getchar?
10. Що виконує функція puts?
11. Що виконує функція sscanf?
12. Що виконує функція strcat?
13. Що виконує функція strcmp?
14. Що виконує функція memcpy?
15. Що виконує функція memmove?
16. Що виконує функція memcmp?
17. Що виконує функція memchr?

Контрольні запитання
1. Скопіюйте стрічку, яка зберігається в масиві s2 в масив s1.
2. Введіть з клавіатури стрічку тексту масив s1 не використовуючи функцію scanf.
3. Визначте довжину стрічки в s1 і виведіть результат.

Література

1. Касаткин А.И., Вальвачев А.Н. Профессиональное программирование на языке Си: От Turbo C Borland C++. –Мн.: Выш. школа, 1992. – 240 с.
2. Подбельский В.В. Язык Си++: Учебн. пособие. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 1996. - 590 с.
3. Березин Б.И., Березин С.Б. Начальный курс С и С++. – М.: ДИАЛОГ-МИФИ, 1998. – 288 с.
4. Уинер Р. Язык Турбо Си. / Перев. с англ. – М.: Світ, 1991. –384 с.
5. Романовская и др. Программирование в среде Си для ПЭВМ ЕС. – М.: Финансы и статистика, 1991.
1. Кнут Д. Искусство программирования на ЭВМ. т.1, Основные алгоритмы. / Пер. с англ. – М: Мир, 1976.
2. Кнут Д. Искусство программирования на ЭВМ. т.2, Получисленные алгоритмы. / Пер. с англ. – М: Мир, 1976.
3. http://void.net.ua/The_C_Programming_Language.html.
4. http://publications.gbdirect.co.uk/c_book/.
5. http://www.scribd.com/doc/16306895/Draft-ANSI-C-Rationale
6. http://www.cplusplus.com/doc/tutorial/

Додаткова література

1. Берри Р., Микинз Б. Язык Си. Введение для программистов. / Пер. с англ. – М.: Финансы и статистика, 1988.
2. Джехани Н. Программирование на языке Си. / Пер. с англ. – М.: Радио и связь, 1988.
3. Керниган Б., Ритчи Д. Язык программирования Си. / Пер. с англ. – М.: Финансы и статистика, 1992.
4. Техника программирования на Turbo C.– М.: И.В.К. – Софт, 1991.
5. Языки программирования Ада, Си, Паскаль. Сравнение и оценка. / Под ред. А.Фьюэра, Н.Джехани. – М.: Радио и связь.