**МІНІСТЕРСТВО ОСВІТИ і НАУКИ УКРАЇНИ, МОЛОДІ та СПОРТУ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"**

**ПРОЕКТ2017**

Інформатика. Персональні комп’ютери та основи програмування**.**

**Методичні вказівки**

**до виконання лабораторних робіт**

**для студентів напрямів підготовки**

**напрям 6.050803 "Акустотехніка**", **6.050903 “телекомунікації“ усіх форм навчання**

***Затверджено методичною радою НТУУ „КПІ”***

**КИЇВ, НТУУ „КПІ”**

**2012**

УДК 621.3.011 (075.8)

ББК 31.211я73 Г94

Прикладне програмування у телекомунікаційних системах [Текст]: Метод.вказівки до викон. лаборатор. Робіт для студентів напрямів підготов. 6.50903 „Телекомунікації” усіх форм навчання/ Укладач: Д.В. Тітков. – К.: НТУУ „КПІ”, 2012. -42 с.

*Гриф надано Методичною радою НТУУ „КПІ”*

*(протокол №\_1\_ від \_22\_.\_вересня\_.2011)*

Навчальне видання

**Інформатика. Персональні комп’ютери та основи програмування.**

**Методичні вказівки**

**до виконання лабораторних робіт**

 **для студентів напрямків підготовки**

**6.050903 “телекомунікації“ усіх форм навчання**

**Аудіо-та відео**

Укладач: *Тітков Дмитро Валерійович*

|  |  |
| --- | --- |
| Відповідальний редакторРецензент | *Ю. Г. Савченко,* доктор.техн.наук, проф.*Т.М. Коротун,* канд. физ.мат.наук, доцент. |

*За редакцією укладачів*

*Надруковано з оригінал-макета замовника*

|  |
| --- |
| **ЗМІСТ** |
| ВСТУП …...…………………………………………………………….. |
| ЛАБОРАТОРНА РОБОТА №1 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №2 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №3 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №4 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №5 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №6 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №7 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №8 ……………………………………… |
| ЛАБОРАТОРНА РОБОТА №9 ……………………………………… |

ВСТУП

Cі — універсальна, [процедурна](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), [імперативна](http://uk.wikipedia.org/wiki/%D0%86%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](http://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) загального призначення, розроблена у [1972](http://uk.wikipedia.org/wiki/1972) році [Денісом Рітчі](http://uk.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D1%96%D1%81_%D0%A0%D1%96%D1%82%D1%87%D1%96) у[Bell Telephone Laboratories](http://uk.wikipedia.org/wiki/Bell_Labs) з метою написання на ній [операційної системи](http://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [UNIX](http://uk.wikipedia.org/wiki/UNIX).[http://uk.wikipedia.org/wiki/C\_(%D0%BC%D0%BE%D0%B2%D0%B0\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) - cite\_note-1](http://uk.wikipedia.org/wiki/C_%28%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29#cite_note-1)

Хоча, Сі і було розроблено для написання [системного програмного забезпечення](http://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), наразі вона досить часто використовується для написання [прикладного програмного забезпечення](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F).

Сі імовірно, є найпопулярнішою у світі мовою програмування за кількістю вже написаного на ній програмного забезпечення, доступного під[вільними ліцензіями](http://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) коду та кількості програмістів, котрі її знають. Реалізації компіляторів для мови Сі існують для багатьох [операційних систем](http://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) та [апаратних архітектур](http://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%95%D0%9E%D0%9C). Cі здійснила великий вплив на інші мови програмування, особливо на [C++](http://uk.wikipedia.org/wiki/C%2B%2B), яка спочатку проектувалася, як розширення для Сі, а також на [Java](http://uk.wikipedia.org/wiki/Java) та [C#](http://uk.wikipedia.org/wiki/C_Sharp), які запозичили у Сі синтаксис.

Сі — мінімалістична [мова програмування](http://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Серед її головних цілей: можливість прямолінійної реалізації компіляції, використовуючи відносно простий [компілятор](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%96%D0%BB%D1%8F%D1%82%D0%BE%D1%80), забезпечити низькорівневий доступ до [оперативної пам'яті](http://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0_%D0%BF%D0%B0%D0%BC%27%D1%8F%D1%82%D1%8C), формувати лише декілька інструкцій машинної мови для кожного елементу мови, і не вимагати обширної динамічної підтримки. У результаті, код Сі придатний для більшості [системного програмного забезпечення](http://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), яке традиційно писалося [асемблером](http://uk.wikipedia.org/wiki/%D0%90%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80).

Незважаючи на її низькорівневі можливості, мова проектувалася для машинно-незалежного програмування. Сумісна зі стандартами та машинно-незалежно написана мовою Cі програма, може легко компілюватися на великій кількості апаратних платформ та [операційних систем](http://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) з мінімальними змінами. Мова стала доступною для великої кількості платформ, від вбудованих [мікроконтролерів](http://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80) до [суперкомп'ютерів](http://uk.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80).

**ЛАБОРАТОРНА РОБОТА №1**

**Використання лінійних та розгалужених структур**

**Мета роботи**— оволодіти практичними навичками розробки, програмування обчислювального процесу лінійної та розгалуженої структури. Отримання подальших навиків по тестування програми.

**Теоретичні відомості**

Змінні та константи можна задавати буквами та числами, першою літерою повинна бути буква. Великі (прописні) та малі (рядкові) букви відрізняються, так буква «х» та «Х» - два різних імені. Зазвичай в програмах на Сі малими буквами позначають змінні, а великими – константи.

В Сі існує лише декілька базових типів:

* Char – одиничний байт, який може містити одну літеру з допустимого набора літер;
* Int – ціле, зазвичай відображається на природній уяві цілих в машині;
* Float – число з плаваючою крапкою одинарної точності;
* Double – число з плаваючою крапкою подвійної точності.

Бінарними арифметичними операторами є +, -, \*, /, а також оператор взяття модуля %. Операторами відношення є >, >=, <, <=.

Функція pow(х,а) підносить число х в степінь а.

Функція exp(x) підносить число е в степінь х.

Функція printf() виводить на екран те, що знаходиться у дужках. При записі printf("rozvjazok 1 %f \n",y”), функція виведе на екран rozvjazok 1 і те число y, яке ми отримаємо у розрахунках самої програми. «\n» означає, що результат буде виводитись з нового рядка.

Знак «=» - присвоєння. Наприклад, запис «а=5» означає, що ми присвоїли змінній а значення 5.

Алгоритм – деяка конкретна послідовність правил, однозначно визначаючих процес перетворення попередніх та проміжних даних в результат вирішення завдання. Для виконання першого завдання нам доведеться використати лінійний алгоритм, для вирішення другої задачі – розгалужений, для третього завдання – циклічний.

Лінійним алгоритмом називають такий алгоритм, дії якого виконуються у заданому програмістом порядку одна за одною.

Приклад

a = 3

b = 5

*S*

початок

кінець

S = a\*b

#include <stdio.h>

int main()

{

 float a = 3, b = 5, S;

 S = a \* b;

 printf("s = %f", S);

 return 0; }

### Алгоритм зветься розгалуженим, якщо послідовність виконання кроків алгоритму змінюється в залежності від деяких умов.

### Умова - це логічний вираз, який може приймати два значення: "так" - якщо умова вірна або "ні" - якщо умова не вірна.

Конструкцію if-else використовують, щоб направити програму тим чи іншим шляхом виконання. В загальному вигляді її можна записати так:

if (вираз-умова)

 інструкція

else

 інструкція-else

Зрозуміло, що інструкція може бути як проста так і складна, представлена блоком.

Програма піде шляхом інструкція тоді, й тільки тоді, коли вираз-умова є істинним висловом. Інакше програма попаде в секцію else, якщо вона присутня. Секція else не є обов'язковою.

### Будь-яка умова складається з трьох частин: ліва частина, знак порівняння, права частина.

Приклад

### Як приклад використання розгалужених алгоритмів створимо програму для перевірки коректності введення даних при розрахунку квадратного кореня. Блок-схема цього алгоритму матиме наступний вигляд:

x < 0

t = 2,2

*y*

початок

кінець

х >= 0



Так

Ні

Код програми:

### #include <stdio.h>

### #include <math.h>

### int main()

### {

###  float x, y;

###  x = 9;

###  if (x >= 0)

###  {

###  y = sqrt(x);

###  printf("y = %f", y);

###  }

###  else

###  printf("x < 0");

###  return 0;

### }

**Робоче завдання**

ЗАВДАННЯ А. Знайти значення змінних, вказаних у таблиці (варіант задається викладачем), по заданим формулам для обчислення і набором вихідних даних. На друк вивести значення вхідних початкових даних і результати обчислень, супроводжуючи вивід вихідних змінних.

Таблиця 1.1— Варіанти завдань

|  |  |  |
| --- | --- | --- |
| Варіант завдання | Формули для обчислення | Значення початкових даних |
| 1 |  | x=1,426y=-1,220z=3,5 |
| 2 |  | x=1,825y=18,225z=-3,298 |
| 3 |  | x=0,335y=0,025 |
| 4 |  | a=-0,5b=1,7t=0,44 |
| 5 |  | a=1,5b=15,5x=-2,9 |
| 6 |  | a=16,5b=3,4x=0,61 |
| 7 |  | a=0,7b=0,05x=0,5 |
| 8 |  | a=1,1b=0,004x=0,2 |
| 9 |  | m=2; c=-1t=1,2; b=0,7 |
| 10 |  | a=3,2b=17,5x=-4,8 |
| 11 |  | a=10,2; b=9,2x=2,2; c=0,5 |
| 12 |  | a=0,3b=0,9x=0,61 |
| 13 |  | a=0,5b=3,1x=1,4 |
| 14 |  | a=0,5b=2,9x=0,3 |
| 15 |  | m=0,7; x=1,7a=0,5; b=1; c=2,1 |

ЗАВДАННЯ Б. Знайти значення функції заданої в таблиці 1.2 (у відповідності з варіантом завдання). Вивести значення ввідних початкових даних і результат обчислення значення функції, супроводжуючи найменування змінних.

Для ЗАВДАННЯ А і ЗАВДАННЯ Б створити окремі програми.

Таблиця 1.2 — Варіанти завдань

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант завдання | Функція | Умова | Початкові дані |
| 1 |  |  | a=-0,5b=2 |
| 2 |  |  | a=1,5 |
| 3 |  |  | a=2,8b=-0,3c=4 |
| 4 |  |  | a=1,65 |
| 5 |  |  | a=2,3 |
| 6 |  |  | a=2,5 |
| 7 |  |  | b=1,5 |
| 8 |  |  | - |
| 9 |  |  | a=20,3 |
| 10 | \*\*\* верхню формулу читати як ( ln3(x) + x2 ) … |  | t=2,2 |
| 11 |  |  | a=2,6b=-0,39 |
| 12 |  |  | a=0,9 |
| 13 |  |  | a=2,1b=1,8c=-20,5 |
| 14 |  |  | a=0,3n=10 |
| 15 |  |  | a=2,5b=0,4 |

**Контрольні запитання**

1. Які типи величин використовуються в мові програмування?
2. Вказати діапазон значень величин цілого і дійсного типів.
3. Які назви змінних допустимі в програмі? Як задати тип змінної в програмі?
4. Вказати назви стандартних функцій для обчислення ,, sin(x), cos(x), ln(x), │x│.
5. Чи можливо в якості операнда в арифметичному виразі використовувати: а) ім`я масиву; б) ім`я стандартної функції, наприклад sin(y); в) ім`я символьної змінної або змінною логічного типу?
6. Назвати послідовність дій при виконанні арифметичного оператора присвоєння. Чи допустиме використання величин різних типів в арифметичному виразі?
7. Написати арифметичний оператор присвоєння для обрахунку значення р(х)=(((а5х+а4)х+а3)х+а1)х+а0.
8. Вказати старшинство виконання операцій при обрахунку арифметичного виразу.
9. Вказати засоби, наявні в мові програмування для управління розміщенням даних у рядку. Як організувати вивід значень, супроводжуючи виведене числове значення найменуванням змінної? Як організувати пропуск одного, двох рядків при виводі?
10. Як вибрати значення вихідних даних для тестового варіанта рахунку?
11. Перерахувати дії, які реалізуються при виконанні умовного оператора.
12. Які дії виконуються оператором переходу?
13. Що таке обчислювальній процес структури, що розгалужується? Як організувати розгалуження обчислень: а) на 2 гілки; б) на 3 гілки?
14. Скласти послідовність операторів для обчислення величини z=0, якщо x<-2; z=1, якщо -2≤x≤2; z=-1, якщо x>2/
15. Для чого необхідно при відладці програми тестувати всі гілки алгоритму?

**ЛАБОРАТОРНА РОБОТА №2**

**Використання циклічних структур**

**Мета роботи**— оволодіти практичними навичками розробки, програмування обчислювального процесу циклічної структури. Отримання подальших навиків по тестування програми.

**Теоретичні відомості**

Цикл — різновид керуючої конструкції, призначена для організації багаторазового виконання набору [інструкцій](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D1%96%D1%8F_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%29) (команд). Одноразове виконання тіла циклу називається ітераціею. Вираз, що визначає чи буде вчергове виконуватися ітерація, чи цикл завершиться, називається  умовою завершення циклу або умовою продовження (в залежності від того, як інтерпретується його істинність). [Змінна](http://uk.wikipedia.org/wiki/%D0%97%D0%BC%D1%96%D0%BD%D0%BD%D0%B0), в якій зберігається номер поточної ітерації, називається лічильником циклу.

Частинами виконання будь-якого циклу є початкова ініціалізація змінних циклу, перевірка умови виходу, виконання тіла циклу і оновлення змінної циклу на кожній ітерації.

Різновиди циклів:

* Цикл з передумовою — цикл, що виконується доки істинна деяка умова, вказана перед його початком. Ця умова перевіряється до початку виконання тіла циклу, тому тіло може бути не виконане жодного разу (якщо умова з початку хибна). Здійснюється за допомогою інструкції while. Приклад:

while(<умова>)

}

<тіло циклу>

}

* Цикл з післяумовою — цикл, в якому умова перевіряється після виконання тіла циклу. Звідси випливає, що тіло циклу завжди виконується хоча б один раз. Здійснюється за допомогою конструкції do-while. Приклад:

do

}

<тіло циклу>

}

while(<умова>)

Цикл завершується, коли умова хибна.

* Цикл з лічильником — цикл, в якому деяка змінна змінює своє значення від заданого початкового значення до кінцевого значення з деяким кроком, і для кожного значення цієї змінної тіло циклу виконується один раз. Реалізується оператором for, в якому вказується лічильник (так звана «змінна циклу»), потрібна кількість проходів і крок, з яким змінюється лічильник. Цикл for, незважаючи на синтаксичну форму циклу з лічильником, в дійсності є циклом з передумовою. Приклад:

for (i=0; i<10; i++)

{

<тіло циклу>

}

фактично являє собою інший варіант запису конструкції:

i=0

while(i<10)

}

<тіло циклу>

i++;

}

Приклад

Як приклад використання циклічних структур створимо програму для знаходження значень функції , для цілих  Блок-схема цього алгоритму матиме наступний вигляд:

початок

кінець



x = 4; x <= 9; x=x+1

y

 x=10

x, y

Код програми:

#include<math.h>

#include<stdio.h>

#include<conio.h>

int main()

{

float x, y;

for(x=4; x<=9; x++)

{

y=pow(2,x);

printf("%d ",y);

}

 getch();

 return 0;

}

**Робоче завдання**

Модифікувати програму ЗАВДАННЯ Б лабораторної роботи №1 таким чином, щоб значення функції обраховувалось багаторазово із заданим кроком у заданому діапазоні. Організувати вивід значення аргументу і обрахованого значення функції у вигляді таблиці.

Таблиця 2.1 — Варіанти завдань

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Варіант завдання | Діапазон і крок зміни аргументу |  | Варіант завдання | Діапазон і крок зміни аргументу |  | Варіант завдання | Діапазон і крок зміни аргументу |
| 1 |  |  | 6 |  |  | 11 |  |
| 2 |  |  | 7 |  |  | 12 |  |
| 3 |  |  | 8 |  |  | 13 |  |
| 4 |  |  | 9 |  |  | 14 |  |
| 5 |  |  | 10 |  |  | 15 |  |

**Контрольні питання**

1. Вказати послідовність дій, які виконуються при організації циклічних фрагментів програми із заданим числом повторів.
2. Різновиди циклів.
3. Перерахувати можливі способи організації циклу із заданим числом повторів в мові програмування Сі.
4. Вказати значення та правила організації циклу.

**ЛАБОРАТОРНА РОБОТА №3 (ЗАВДАННЯ А)**

**Обробка одновимірних масивів**

**Мета роботи**— оволодіти практичними навиками роботи з масивами, особливостями їх вводу та виводу, набуття подальших навиків з організації програм циклічної структури.

**Теоретичні відомості**

Масив – одина з най­більш відомих структур даних. Під масивом в мові Сі розу­міють набір даних одного і того ж типу, зібраних під одним ім'ям. Кожний елемент масиву визначається ім'ям масиву і порядковим номе­ром елемента, який називається індексом. Індекс в мові Сі завжди ціле число.

Частіше за все використовуються одновимірні масиви:

тип <ім'я масиву> [розмір] ;

Розмір масиву в мові Сі може задаватися константою або констан­тним виразом. Не можна задати масив змінного розміру. Для цього існує окремий механізм, званий динамічним виділенням пам'яті.

У мові Сі індекс завжди починається з нуля. Коли ми говоримо про перший елемент масиву, то маємо на увазі елемент з індексом 0. Якщо ми оголосили масив int a[100] це означає, що масив містить 100 елементів від а[0] до а[99]. Для одновимірного масиву легко підрахувати, скільки байт в пам'яті бу­де займати цей масив:

кільк. байтів = <розмір базового типу> \* <кільк. елементів>.

У мові Сі під масив завжди виділяється безперервне місце в опе­ративній пам'яті.

У мові Сі не перевіряється вихід індексу за межі масиву. Якщо масив а[100] описаний як цілочисельний масив, що має 100 елемен­тів, а ви в програмі вкажете а[200], то повідомлення про помилку не буде видане, а як значення елемента а[200] буде видано деяке число, що займає відповідні 2 байти. Можна визначити масив будь-якого визначеного раніше типу, наприклад:

unsigned arr[40], long double al[1000], char ch[80].

Перед тим як використати масив, необхідно присвоїти значення його елементам. У Сі масиви не ініціалізуються і не обнулюються автоматично. Елементам масиву можна задати початкові значення трьома способами: ініціалізацією, присвоєнням або введенням. При описі масиву може бути виконана ініціалізація елементів масиву.

Є два методи ініціалізації:

1. *ініціалізація за замовчуванням.* Якщо не проводиться ініціалізація елементів масиву, то всі елементи статичних та зовнішніх масивів ініціалізуються компілятором нулями, а елементи автоматичних і регістрових масивів визначені на неочищену пам’ять;
2. *явна ініціалізація елементів*. Після опису масиву можна
записати список початкових значень масиву, які беруться у фігурні дужки.

Існують дві форми явної ініціалізації:

1. явне зазначення числа елементів масиву та список початкових значень, можливо, з меншим числом елементів.

<тип даних> <ім’я масиву> [розмірність N]...[розмірність 1] = {<список значень>};

Наприклад, ініціалізація зовнішнього масиву з 10 елементів:

int array [10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

Якщо оголошений масив b з 10 елементів int b [10] = {1, 2, 3, 4}, то перші чотири елементи масиву ініціалізувати числами 1, 2, 3 і 4. Значення інших шести елементів або дорівнює 0, якщо масив зовнішній чи статистичний, або не визначено, якщо масив автоматичний чи регістровий.

Завдання в списку початкових значень більшого числа значень, ніж є елементів у масиві, є синтаксичною помилкою;

2) без явного зазначення елементів масиву, тільки зі списком початкових значень. Компілятор визначає число елементів масиву за списком ініціалізації.

<тип даних> <ім’я масиву> [ ][розмірність N–1]...[розмірність 1] =
 {<список значень >};

Наприклад, int array [ ] = {1, 2, 3, 4};

У результаті створюється масив з чотирьох елементів, і ці елементи отримують початкові значення зі списку ініціалізації.

Масиву символів можна задавати початкові значення, використовуючи рядкову константу. Наприклад, оголошення

char c[ ] = {"ABCD"};

присвоює елементам масиву *с* початкові значення як окремі символи рядка “ABCD”. Розмір масиву *с* визначається на основі довжини рядка плюс нульовий символ закінчення рядка, таким чином масив містить п’ять елементів. Символьний масив можна задати окремими символьними константами. Попереднє оголошення еквівалентне наступному:

char c [ ] = {'A', 'B', 'C', 'D', '\0'};

Використовуючи операцію присвоєння, можна в програмі присвоїти значення кожному елементу масиву або, використовуючи оператор циклу і функції введення, можна вводити значення елементів з клавіатури. Ці способи ініціалізації масиву будуть показані нижче.

Доступ до елементів масиву може виконуватися за допомогою операції індексації або за допомогою механізму вказівників. При використанні операції індексації для посилання на необхідний елемент вказується його номер у масиві, вміщений у [ ].

У мові Сі дозволяється лише поелементне звертання до масиву, при якому поточне значення може бути задане константою, змінною чи виразом. Наприклад,

int a[9]; a[0] = 0; a[1] = 0; a[2] = 0; a[3] = 0; a[4] = 0; a[5] = 0; a[6] = 0; a[7] = 0; a[8] = 0;

У даному прикладі обнуляються елементи масиву a.

Інший спосіб доступу до елементів масиву — використання механізму вказівників. Ім’я масиву, що використовується без наступних [ ], являє собою адресу початку масиву. Оскільки ім’я масиву — це вказівник-константа на перший байт першого елемента масиву, то, використовуючи операцію отримання значення за адресою (\*), можна виконати доступ до будь-якого елемента масиву. Тобто будуть еквівалентними запис до посилання на i-й елемент масиву array array[i] і \*(array+i).

Нехай є опис масиву int a[5], який містить 5 елементів: a[0], a[1], a[2], a[3], a[4]. Адреса і-го елемента масиву дорівнює сумі адреси початкового елемента масиву і зсування цього елемента на і одиниць від початку масиву. Якщо pa — це вказівник на ціле int \*pa; , то після виконання оператора pa = &a[0]; pa містить адресу елемента a[0]. Вираз pa+1 вказує на наступний елемент, pa+i вказує на i-й елемент масиву, тобто є адресою a[i], тоді \*(pa+i) є вмістом i-го елемента. Оскільки ім’я масиву ототожнюється з адресою його першого елемента, то оператор pa = &a[0]; еквівалентний оператору pa = a; , тому будуть еквівалентними запис до посилання на i-й елемент масиву a[i] і \*(а+i). Будь-який масив і індексний вираз можна представити за допомогою вказівника. В той же час між ім’ям масиву й відповідним вказівником є істотна відмінність. Треба пам’ятати, що вказівник — це змінна, а ім’я масиву — константа. Тому pa = a; і pa++; допустимі операції. Оператори вигляду а = pa; а++; використати не можна, оскільки значення константи постійне і не може бути змінене.

Якщо до вказівника додається ціле, компілятор автоматично масштабує ціле, множачи його на число байтів, відповідне типу, зазначеному в оголошенні вказівника. Розміщення масиву а в пам’я­ті подано на рис. 5.1, якщо він розташовується з адреси 1000.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1000 | 1002 | 1004 | 1006 | 1008 |
| \*ра | \*(pa+1) | \*(pa+2) | \*(pa+3) | \*(pa+4) |
| a[0] | a[1] | a[2] | a[3] | a[4] |
| \*а | \*(а+1) | \*(а+2) | \*(а+3) | \*(а+4) |

Рисунок 5.1. Розміщення масиву в пам’яті

Тоді вираз ра + 3 або а + 3 дає значення адреси 1000 + 3\*si­zeof(int) = 1000 + 3\*2 = 1006. Взявши значення за цією адресою, можна набути значення четвертого елемента масиву, тобто \*(ра+3) або \*(а+3). Обидва способи еквівалентні.

Приклад

Розробимо алгоритм для підрахунку кількості ненульових значень масиву А[10].

Ні

*p*

початок

кінець

p = p + 1

Ai = 0

N = 10, A[N]

i = 0; i <= N; i=i+1

Так

 i=N+1

Код програми:

#include <stdio.h>

int main()

{

 int N = 10;

 int p = 0;

 int A[N] = {5, -3, 0, 3, 9, -2, 0, 2, 1, 4};

 for (int i = 0; i < N; i++)

 if (A[i] == 0) p++;

 printf("%d", p);

 return 0;

}

**Робоче завдання**

Створити масив цілих чисел. Обробити масив у відповідності до варіанту завдання вказаного у таблиці. Масив створити з елементами цілого типу. Заповнюючи масив, робити це таким чином щоб у масиві знаходилися елементи які задовольняють умову з стовпчика „Умови і обмеження” а також за цими межами. Виконуючи завдання з колонки „Дії” враховувати тільки ті елементи які підходять до умови з колонки „Умови і обмеження”. Перевірити правильність виконання програми за допомогою тестового варіанту.

Таблиця 4.1 — Варіанти завдань

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант завдання | Масив | Дії | Умови і обмеження |
| 1 | **Х(10)** | Обрахувати суму і кількість елементів  | 0 ≤ X[i] ≤ 4 |
| 2 | **А(15)** | Обрахувати середнє арифметичне значення елементів  | A[i] > 0 |
| 3 | **X(12)** | Переписати елементи масиву Х в масив Y і підрахувати їх кількість | -2 ≤ X[i] ≤ 2 |
| 4 | **B(14)** | Визначити максимальний елемент масиву В і його порядковий номер | B[i] >0 |
| 5 | **С(10)** | Знайти мінімальний елемент масиву С і його номер | C[i] <0 |
| 6 | **D(17)** | Знайти максимальний і мінімальний елемент масиву D і поміняти їх місцями. |  |
| 7 | **Y(12)** | Обрахувати середнє геометричне елементів | Y[i] >0 |
| 8 | **Z(13)** | Розташувати у масиві R спочатку додатні а потім від’ємні елементи масиву Z. |  |
| 9 | **N(15)** | Визначити суму елементів масиву N, кратних трьом. |  |
| 10 | **X(10)** | Обрахувати суму і кількість елементів масиву | X[i]>0 |
| 11 | **A(12)** | Знайти середнє геометричне елементів  | A[i]> 0 |
| 12 | **X(14)** | Переписати в масив Y підряд додатні елементи масиву Х. | X[i] >0 |
| 13 | **X(16)** | Переписати підряд в масив Y додатні і в масив Z від’ємні елементи масиву Х. |  |
| 14 | **B(12)** | Визначити максимальний елемент масиву В і його порядковий номер. | B[i]< 0 |
| 15 | **С(16)** | Визначити мінімальний елемент масиву С і його порядковий номер. | -2 ≤ C[i] ≤ 2 |

**Контрольні запитання**

1. Вказати особливості програм, в яких використовуються масиви.
2. Які оператори мови можна використовувати для опису масивів?
3. В чому полягає особливість організації цикла при обробці масивів?
4. В чому полягає особливість використання прийомів програмування при обробці масивів?
5. Вказати особливості вводу і виводу масивів.

**ЛАБОРАТОРНА РОБОТА №3 (ЗАВДАННЯ Б)**

**Робота з багатовимірними масивами**

**Мета роботи**— оволодіти практичними навиками роботи із масивами, особливостями їх вводу і виводу, отримання подальших навиків по організації програм циклічної структури із використанням прийомів програмування.

**Теоретичні відомості**

Сі підтримує багатовимірні масиви. Найпростішим видом багатовимірного масиву є двовимірний масив, який можна представити як масив одновимірних масивів. Двовимірний масив являє собою матрицю, де перший індекс відповідає за рядок, а другий за стовпець. Кожна розмірність масиву вміщується в окремі квадратні дужки. Багатовимірний масив оголошується таким чином:

<тип даних> <ім’я масиву> [розмірність N]...[розмірність 2] [розмірність 1];

Елементи багатовимірного масиву зберігаються в пам’яті в порядку зростання найправішого індексу, тобто за рядками. Це означає, що правий індекс змінюється швидше лівого, якщо переміщатися масивом у порядку розташування елементів в пам’яті.

При ініціалізації багатовимірних масивів можна додати фігурні дужки навколо кожного вимірювання. Наприклад, int d [2][3] = {{1, 2, 3},{4, 5, 6}};

Якщо даних рівно стільки, скільки має бути при заповненні всіх елементів, то фігурні дужки в списку можуть бути опущені, тоді наступні два записи еквівалентні:

int f [2][2] = {{1, 2}, {3, 4}}; int f [2][2] = {1, 2, 3, 4};

Для багатовимірних масивів необхідно визначити всю розмірність, крім найлівішої, компілятор визначає число елементів за числом значень у списку ініціалізації. Наприклад,

int array [ ] [3] = {0, 1, 2, 3, 4, 5}; тобто кожен рядок міститиме три елементи: нульовий — 0, 1, 2; перший — 3, 4, 5. Якщо потрібно проініціалізувати не всі елементи, в списку ініціалізації використовуються фігурні дужки, наприклад:

int array [ ][3] = {{0}, {3, 4}};

m [0][0] дорівнюватиме 0, m [1][0] — 3, m [1][1] — 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Стовпець 0 | Стовпець 1 | Стовпець 2 | Стовпець 3 |
| Рядок 0 | а[0][0] | а[0][1] | а[0][2] | а[0][3] |
| Рядок 1 | а[1][0] | а[1][1] | а[1][2] | а[1][3] |
| Рядок 2 | а[2][0] | а[2][1] | а[2][2] | а[2][3] |

 Рисунок 6.1. Двовимірний масив з трьома рядками і чотирма стовпцями

Приклад

 Як приклад виконання операцій над матрицями розробимо алгоритм для виведення на екран таблички множення розмірами 10х10. Блок-схема такого алгоритму матиме наступний вигляд:



початок

кінець



i = 0; i < 10; i=i+1

j = 0; j < 10; j=j+1

 i=11

 j=11

Код програми:

#include <stdio.h>

#include <conio.h>

int main()

{

 clrscr();

 int A[10][10], i, j;

 for (i = 0; i < 10; i++)

###  {

###  for (j = 0; j < 10; j++)

###  {

###  A[i][j] = (i + 1) \* (j + 1);

###  printf("%4d", A[i][j]);

###  }

###  printf("\n");

###  }

###  return 0;

**}**

**Робоче завдання**

Створити матрицю з цілими числами. Обробити матрицю у відповідності з варіантом, вказаного у таблиці. Вивести на друк результати і вихідну матрицю у загальноприйнятому вигляді. Перевірити правильність виконання програми за допомогою тестового завдання.

Таблиця 5.1 — Варіанти завдань

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант завдання | І'мя матриці та розміри | Дії | Умова та обмеження |
| 1 | **A(10;15)** |  Обрахувати і запам’ятати суму і число додатніх елементів кожного стовпця матриці. Результати роздрукувати у вигляді двох рядків. | aij>0  |
| 2 | **A(N;M)** |  Обрахувати і запам’ятати суми і числа елементів кожного рядка матриці. Результат роздрукувати у вигляді двох стовпців. | N≤20M≤15 |
| 3 | **B(N;M)** |  Обрахувати суму і число елементів матриці, що знаходяться під головною діагоналлю і на ній. |  N≤12 |
| 4 | **V(15;10)** | Впорядкувати за збільшенням елементи кожного рядка матриці. |  |
| 5 | **C(N;N)** |  Обрахувати суму і число додатних елементів матриці, що знаходяться над головною діагоналлю. |  cij>0N≤12 |
| 6 | **D(K;K)** | Записати на місце від’ємних елементів матриці нулі, і вивести її на друк у загальноприйнятому вигляді.   |  K≤10 |
| 7 | **D(10;10)** |  Записати на місце від’ємних елементів матриці нулі, а на місце додатних – одиниці. Вивести на друк нижню трикутну матрицю у загальноприйнятому вигляді. |  |
| 8 | **F(N;M)** |  Знайти у кожному рядку матриці максимальний і мінімальний елементи, і розмістити їх на місце першого і останнього елемента рядка відповідно.  |  N≤20M≤15 |
| 9 | **F(10;10)** |  Транспонувати матрицю і вивести на друк елементи головної діагоналі і діагоналі розміщеної під головною. Результати розмітити у двох рядках. |   |
| 10 | **N(10;10)** |  Для цілочислової матриці знайти для кожного рядка число елементів кратних 5, і найбільший з отриманих результатів.  | nij/5\*5=nij |
| 11 | **N(10;10)** | Із додатних елементів матриці N сформувати матрицю М (10,КМАХ), розміщуючи їх у рядках матриці підряд, де КМАХ – максимальне число додатних елементів рядка матриці N. Записати нулі на місце відсутніх елементів. Роздрукувати обидві матриці |  |
| 12 | **P(N;N)** |  Знайти у кожному рядку найбільший елемент, і поміняти його з місцями елементом головної діагоналі.  |  N≤15 |
| 13 | **R(K;N)** | Знайти найбільший і найменший елементи матриці, і поміняти їх місцями.  | N≤20M≤10  |
| 14 | **S(25;8)** |  Ввести початкові дані в перші 24 рядка і перші 7 стовпців. Обрахувати середнє арифметичне значення елементів кожного рядка і записати його у 8-й стовпчик, а також середнє арифметичне кожного стовпчика і записати його у 25-й рядок.  |   |
| 15 | **T(N;M)** |  Знайти рядок з найбільшою і найменшою сумую елементів. Вивести на друк знайдені рядки і суми їх елементів. |  N≤20M≤15 |

**Контрольні запитання**

1. Вказати основні правила організації вкладених циклів.
2. Вказати способи виходу з внутрішнього циклу.
3. Як організувати вивід матриці в загальноприйнятому вигляді?
4. Як організувати вивід нижньої трикутної матриці в загальноприйнятому вигляді?
5. Як організувати вивід матриці розміром N\*M елементів?

 **ЛАБОРАТОРНА РОБОТА №4**

**Програмування з використанням підпрограм користувача**

**Мета роботи**— оволодіти навиками алгоритмізації і програмування задач з використанням підпрограм користувача, оволодіти навиками написання підпрограм і звертання до них.

**Теоретичні відомості**

Під бібліотечними підпрограмами розуміють підпрограми, які були створені раніше та розміщені в спеціальних програмних блоках, що називаються модулями або бібліотеками. До таких підпрограм можна звернутися у власній програмі. Для цього достатньо підключити модуль, де вона описана.

Програмісти розрізняють два види підпрограм: *процедури та функції*. Цей поділ умовний у мові С.

*Функція* – це підпрограма, яку викликають, щоб виконати якісь розрахунки чи перевірки. Коли вона закінчує роботу, то повертає управління програмі, з якої викликалась і передає їй результати розрахунків.

Наприклад,

int func1(int x)

{

 x=x+1;

 return x;

 }

*Процедура* – це підпрограма, яку викликають, щоб виконати деякі дії, але від неї не вимагається повертати основній програмі певні значення. У мові С умовно вважається, що процедур немає, але функція типу *void* виконує відомі нам функції процедури, не повертаючи ніякого значення підпрограмі. Наприклад,

void func2(int &x)

{

 x=x+1;

 }

Приклад

Припустимо, що нам потрібно розробити програму для знаходження факторіалу числа. Для вирішення цієї задачі напишемо функцію. Блок-схема алгоритму матиме наступний вигляд:

початок

кінець

fact(a)

a

вхід

вихід

i = 1; i <= n; i=i+1

i

r = r \* i

r = 1

fact = r

 Функція fact

Код програми:

### #include <conio.h>

### #include <stdio.h>

### int fact(int n)

### {

###  int i, r = 1;

###  for (i = 1; i <= n; i++)

###  r \*= i;

###  return r;

### }

### int main()

### {

###  clrscr();

###  int a = 4;

###  printf("%d", fact(a));

###  return 0;

### }

Припустимо, що нам потрібно заповнити декілька масивів довільними числами. Для вирішення цієї задачі напишемо процедуру. Блок-схема алгоритму матиме наступний вигляд:

початок

кінець

 Fill( A, 10 )

 Fill( B, 20 )

вхід

вихід

i = 0; i <= n; i=i+1

i

M[ i ]

 Процедура Fill

Код програми:

### #include <stdio.h>

### void Fill(int \*M, int n)

### {

###  int i;

###  for (i = 0; i <= n; i++)

###  M[i] = rand() % 10;

### }

### int main()

### {

int A[10], B[20];

Fill( A, 10 );

Fill( B, 20 );

### return 0;

### }

**Робоче завдання**

Скласти програму яка реалізує задачу вказану у Табл. 6.1. На друк вивести значення вхідних початкових даних і результати обчислень.

Таблиця 6.1 — Варіанти завдань

|  |  |  |
| --- | --- | --- |
| Варіант завдання | Умова задачі | Використовувані підпрограми |
| 1 | Розрахувати більші корені квадратних рівнянь   | Всі корені дійсні, повертати найбільний корінь з двох |
| 2 | Підрахувати число точок, що знаходяться всередині околу радіусом *r* з центром в початку координат; координати задані масивами X(15), Y(15) | Відстань точки від початку координат розрахувати в підпрограмі |
| 3 | Знайти периметри трикутників, заданих координатами їх вершин  | Довжину сторони трикутника розрахувати в підпрограмі |
| 4 | Підрахувати число точок, що знаходяться всередині околу радіусом *r* з центром в точці з координатами (2, 2); координати задані масивами X(17), Y(17) | Відстань точки від початку координат розрахувати в підпрограмі |
| 5 | Розрахувати , де, ,  -- об’єми шарів із радіусами  відповідно |  рахувати в підпрограмі |
| 6 | Розрахувати суми додатних елементів масивів X(12), Y(15), Z(17) | У підпрограмі обробляти один масив |
| 7 | Розрахувати середнє арифметичне додатних елементів для масивів A(10), B(15), C(20) | У підпрограмі обробляти один масив |
| 8 | Підрахувати кількість елементів матриць X(10, 15) i Y(20, 12), що задовольняють умовам  і  | У підпрограмі обробляти один масив |
| 9 | Розрахувати суму додатних елементів кожного рядку для матриць А(10, 12) і В(15, 10) | У підпрограмі обробляти один масив |
| 10 | Розрахувати , де  і  -- найменші елементи масивів X1(14), X2(16) | У підпрограмі обробляти один масив і повертати мінімум з елементів |
| 11 | Розрахувати суму елементів головних діагоналей матриць A(N, N) i B(M, M) | У підпрограмі обробляти один масив   |
| 12 | Розрахувати , де  -- сума додатних елементів масиву X(14);  -- сума від’ємних елементів масиву Y(16)  | У підпрограмі обробляти один масив, підпрограма повинна бути універсальною |
| 13 | Підрахувати число нульових елементів для матриць A(N, M) i B(M, N) | У підпрограмі обробляти один масив  |
| 14 | Розрахувати суми елементів нижніх трикутних матриць для матриць А(15, 15) і В(20, 20) | У підпрограмі обробляти один масив |
| 15 | Знайти число додатних елементів до першого від’ємного в масивах X(24), Y(17), Z(13)  | У підпрограмі обробляти один масив |

**Контрольні запитання**

1. Вказати структуру опису підпрограми БСП.
2. Вказати переваги, які отримують користувачі, від використання БСП.
3. Перерахувати способи зберігання матриць, які використовуються в БСП.
4. Вказати особливості використання матриць в БСП та способи їх зберігання в пам`яті машини.
5. Вказати, як передаються вирази в підпрограмі БСП.
6. Вказати, як узгоджуються формальні та фактичні параметри підпрограм БСП.

**ЛАБОРАТОРНА РОБОТА №5**

**Обробка символьних даних**

**Мета роботи**— оволодіти навиками алгоритмізації і програмування задач, що обробляють символьні дані; вводу і виводу символьних даних, їх обробки; використання функцій обробки символьних даних.

### Теоретичні відомості

Стринги є одним з найбільш корисних та важливих типів даних мови Сі. Символьний рядок (стринг) - це масив символів, замкнений у лапки ("). Він має тип char. Задати рядок можна трьома способами:

* Об’явивши рядок і її розмір, а потім записати в неї текст:

char word[5];

word[5]=”line”;

* Об’явити рядок, не вказавши його довжини:

char str[]=”line”;

* Також рядок можна задати по кожному елементу по черзі:

char str[5]={'l', 'i', 'n', 'e', '\0'}; де \0 – вказує на кінець рядка.

 Нульовий символ (\0) автоматично додається останнім байтом символьного рядка та виконує роль ознаки його кінця. Кількість елементів у масиві дорівнює кількості символів у стрингу плюс один, оскільки нульовий символ також є елементом масиву. Кожна стрингова константа, навіть у випадку, коли вона ідентична іншій стринговій константі, зберігається у окремому місці пам'яті. Якщо необхідно ввести у рядок символ лапок ("), то перед ним треба поставити символ зворотного слешу (\). У стринг можуть бути введені будь-які спеціальні символьні константи, перед якими стоїть символ \.

Прототипи всіх функцій, що працюють з рядками символів, містяться у файлі string.h. Всі функції працюють з рядками, що закінчуються нульовим символом. Ось деякі з них:

strcat(line1, line2) - з'єднати два стринги;

strcpy(line1, line2) - копіювати рядок s2 у рядок s1;

strlen(line) - визначити довжину рядку (кількість символів без нульового символа).

Приклад

Як приклад роботи з символьними даними розробимо алгоритм для знаходження кількості елементів рядка. Блок-схема такого алгоритму:

 char line[]

strlen(line)

початок

кінець

Код програми:

#include<string.h>

#include<stdio.h>

int main()

{

 char line[]="agrg";

 printf("%d",strlen(line));

 getch();

 return 0;

}

**Робоче завдання**

Скласти програму у відповідності з варіанту, вказаного у табл. 8.1. Вивести на друк вхідний рядок та результати у загальноприйнятому вигляді. Перевірити правильність виконання програми за допомогою тестового завдання.

Таблиця 7.1 — Варіанти завдань

|  |  |
| --- | --- |
| Варіант завдання |  Умова задачі |
| 1 | Перевірити, чи є в заданому тексті баланс дужок, що відкриваються та закриваються. |
| 2 |  Для пар, що зустрічаються в заданому тексті поруч розташованих символів вказати, скільки раз зустрічається кожне з таких двобуквених сполучень.  |
| 3 |  Відредагувати речення, видаляючи з нього зайві пробіли, залишаючи тільки по одному пробілу між словами. |
| 4 |  В заданому реченні вказати слово, в якому доля голосних (А, Е, І, О) максимальна. |
| 5 |  Для кожного символу заданого тексту вказати, скільки раз зустрічається у тексті. Повідомлення про один символ не повинно зустрічатися більше, ніж один раз. |
| 6 |  Для кожного слова заданого речення вказати долю голосних. Визначити слово, в якому доля голосних максимальна. |
| 7 | Знайти саме довге симетричне слово заданого речення, наприклад АККА.  |
| 8 | Відредагувати задане речення, замінюючи багатокрапки однією крапкою.   |
| 9 | В заданому реченні знайти саме довге та коротке слово.  |
| 10 |  Із заданого тексту речення вибрати і надрукувати тільки ті символи, які зустрічаються в ньому тільки один раз (у тому порядку, в якому вони зустрічаються в тексті). |
| 11 |  В заданому тексті замінити послідовність символів Х (І) на А (І) и підрахувати число здійснених замін. |
| 12 |  В заданому тексті видалити символ «,» і підрахувати число видалених символів. |
| 13 |  З тексту вибрати числа і записати в масив N. Кількість чисел не більше 10. |
| 14 |  Видалити із тексту символи «ٮ» і підрахувати довжину сформованого тексту. |
| 15 |  В тексті речення замінити символ «ٮ» на символи «,». Кінцеві символи видалити не заміняючи на коми. Визначити довжину речення. Якщо у тексті зустрічається декілька символів «ٮ» підряд, то замість них поставити одну кому. |

**Контрольні запитання**

1. Вказати області використання символьних даних.
2. Вказати способи запису символьних констант.
3. Які операції можна виконувати над символьними даними?
4. Вказати спосіб описання символьних даних.
5. Назвати спеціальні функції, що використовуються для обробки символьних даних.

**ЛАБОРАТОРНА РОБОТА №6**

**Програмування з виводом результатів у вигляді графіків на екран алфавітно-цифрового дисплея**

**Мета роботи**— оволодіти методами виведення результатів у вигляді графіків на екран, практичними навиками та можливостями мови програмування для обробки і виведення символьних даних.

**Теоретичні відомості**

## Дисплей – це пристрій для введення та виведення інформації, який забезпечує візуальне представлення даних у зручній для сприйняття людиною формі.  Алфавітно-цифровий дисплей  – дисплей, у якому передбачена можливість відображення алфавітно-цифрових знаків та спеціальних символів.

Для виводу графіку функції на екран, можна скористатися досить простим алгоритмом. Ввести змінну *h=,* що буде визначати номер положення в рядку, де необхідно зобразити значення функції . Коефіцієнти *k, b* потрібні для масштабування. В циклі до значення *h* в черговому рядку, що відповідає значенню аргументу  і функції  за допомогою оператора *printf* виводимо пустий символ " ". При номеру в рядку, що відповідає *h* друкуємо "\*" і стрибаємо на наступний рядок.

Нагадаємо, що керуюча послідовність функції *printf* має вигляд:

**%** [*ширина*][**.***точність*]*тип*

Приклад

Як приклад виведення даних на алфавітно-цифровий дисплей можна створити програму, яка виводить нулі у непарні та одиниці у парні рядки екрану. Блок-схема такого алгоритму матиме наступний вигляд:

початок

кінець

i % 2

i = 0; i < 10; i=i+1

j = 0; j < 10; j=j+1

 i=11

 j=11

Код програми:

### #include <conio.h>

### #include <stdio.h>

### int main()

### {

###  clrscr();

###  int i, j;

###  for (i = 0; i < 24; i++)

###  for (j = 0; j < 80; j++)

###  printf("%d", i % 2);

###  return 0;

### }

**Робоче завдання**

Вивести на екран алфавітно-цифрового дисплея таблицю значень функції *y=f(x)* та її графік в заданому діапазоні зміни аргументу від а до b з числом точок графіка n.

Таблиця 8.1 — Варіанти завдань

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант завдання | Вид функції  | Діапазон зніни аргументу | Кількість точок графіка n |
| а | b |
| 1 | *sin (x)* | -π/2 | π/2 | 30 |
| 2 | *cos (x)* | 0 | 3π/2 | 40 |
| 3 | *│sin x│+│cos x│* | 0 | π | 40 |
| 4 | *│sin x│-│cos x│* | 0 | π | 40 |
| 5 | *2sin x+3cos x* | - π | + π | 50 |
| 6 | *sin x+cos(2x)* | -π | +π | 50 |
| 7 | *2-cos x* | 0 | 3π/2 | 40 |
| 8 | *sin(2x)1/2+cosx* | 0 | 2π | 50 |
| 9 | *2sin(2x)+1* | - π/2 | + π/2 | 50 |
| 10 | *sin x+cos x-1* | - π | + π | 40 |
| 11 | *(x2+2)1/2* | -3 | 5 | 40 |
| 12 | *10/(1+x2)* | -3 | 3 | 30 |
| 13 | *(x-3)/(x2+2)* | -1 | 4 | 50 |
| 14 | *x\*cos(2x)* | -1 | 4 | 50 |
| 15 | *x2e-│x│* | -1 | 3 | 40 |

**Контрольні запитання**

1. Вказати, за допомогою яких операторів можна зарезервувати пам`ять під символьний масив С, який складається із 100 елементів.
2. Які операції можна виконувати над символьними змінними?
3. Як організувати ввід-вивід символьних змінних?
4. За допомогою яких операторів мови ініціалізовуються початкові значення символьних змінних?
5. Як вибрати масштаби графіка, який викреслюємо, по осям х та у?
6. Чому необхідно затерти символ графіка в символьному масиві при переході до наступного рядка графіка?
7. Як видозмінити програму, щоб викреслювалась сітка графіка, схожа на велику сітку на міліметровому папері?
8. Як видозмінити програму , щоб виконувалось підписування осей х, у та значень аргументу х біля відповідного рядка графіка?
9. Як вивести на друк графік двох функцій від одного аргументу в однакових масштабах? В різних масштабах?

**ЛАБОРАТОРНА РОБОТА №7**

**Обробка файлових структур даних**

**Мета роботи**— оволодіти навичками алгоритмізації і програмування файлових структур даних; проектування структури файлу, вивід даних у файл, зчитування даних із файлу.

**Теоретичні відомості**

**Робоче завдання**

Створити програму для створення та обробки файлу відповідно до завдання вказаного в табл.9.1. Для ЗАВДАННЯ А і ЗАВДАННЯ Б створити окремі програми.

Таблиця 9.1 – Варіанти завдань

|  |  |
| --- | --- |
| Варіант завдання | Умова завдання |
| 1 | 1. Створити файл, що містить інформацію про місячну заробітну платню працівників заводу. Кожен запис містить поля: прізвище працівника, назва цеху, розмір заробітної плати за місяць. Кількість записів довільна.
2. Розрахувати загальну суму витрат за місяць цеху X, а також середньомісячний заробіток працівника цього цеху. Надрукувати для бухгалтерії відомість для нарахування заробітної плати працівникам цього цеху.
 |
| 2 | 1. Створити файл, що містить дані про кількість виробів, зібраних працівниками цеху за неділю. Кожен запис містить поля: прізвище працівника, кількість виробів, що він зібрав щодня протягом неділі; тобто окремо— в понеділок, у вівторок і т. д. Кількість записів довільна.
2. Написати програму, що видає на друк таку інформацію: прізвище працівника та загальну кількість виробів, зібраних їм за тиждень; прізвище працівника, який зібрав найбільшу кількість виробів, і день, коли він досяг найвищої продуктивності праці.
 |
| 3 | 1. Створити файл, що містить дані про кількість виробів категорії А, В, С, зібраних працівниками за місяць. Структура запису містить поля: прізвище працівника, назва цеху, кількість виробів по категоріям, зібраних працівниками за місяць. Кількість записів довільна.
2. Вважаючи заданими значення цін SA, SB, SC за виконану роботу з виготовлення деталі А, В, С відповідно, вивести на друк наступну інформацію:
	* загальна кількість виробів категорій А, В, С, зібраних працівниками цеху X
	* відомість заробітної плати працівників цеху X;
	* середній розмір заробітної плати працівників цього цеху.
 |
| 4 | 1. Створити файл, що містить дані про телефони абонентів. Кожен запис має поля: прізвище абонента, рік установки телефону, номер телефону. Кількість записів довільна.
2. Написати програму, що видає інформацію наступного виду:
* за введеним прізвищем абонента виводиться номер телефону;
* визначається кількість встановлених телефонів з XXXX року. Номер вводиться з терміналу.
 |
| 5 | 1. Створити файл, що містить дані про асортимент іграшок в магазині. Структура запису: назва іграшки, ціна, кількість, вікові рамки(наприклад 2—5, тобто від 2 до 5 років). Кількість записів довільна.
2. Написати програму, в результаті виконання якої виводяться наступні дані:
* назва іграшок, що підходять дітям від 1 до 3 років;
* вартість найдорожчої іграшки та її назва;
* назва іграшки, що не перевищує x гривень і підходить для дитини віком від a до b років. Значення x, a, b ввести з терміналу.
 |
| 6 | 1. Створити файл, що містить відомості про результати сесії студентів І курсу кафедри «ЗТ і РІ». Структура запису: індекс групи, прізвище студента, оцінки з п’яти екзаменів, участь у громадській діяльності («1» — активна участь, «0» — неучасть) . Кількість записів— 30.
2. Написати програму зарахування студентів групи X на стипендію. Студент, що отримав всі оцінки «5» та активно брав участь у громадській роботі, зараховується на підвищену стипендію (доплата 50%), не активно бере участь— доплата 25%. Студенти, які отримали «4» і «5», зараховуються на звичайну стипендію. Студент, який отримав одну оцінку «3», але активно бере учать у громадській роботі, також зараховується на стипендію, в іншому випадку зарахування не відбувається. Індекс грипи вводиться з терміналу
 |
| 7 | 1. Створити файл, що містить відомості про результати сесії студентів. Структура запису: індекс групи, прізвище студента, оцінки з п’яти екзаменів та п’яти заліків («з» — залік, «н» — незалік). Кількість записів — 25.
2. Написати програму, в результаті виконання якої виводяться наступні дані:
	* прізвище студентів, які мають заборгованості, з вказівкою індексів груп та кількістю боргів;
	* середній бал, отриманий кожним студентом групи X, і всієї групи в цілому.
 |
| 8 | * + 1. Створити файл, що містить дані про особисту колекцію книголюба. Структура запису: шифр книги, автор, назва, рік видання, місцезнаходження (номер шафи, полиці і т. д.). Кількість записів довільна.
		2. Написати програму, яка виводить наступні дані:
* місцезнаходження книги автора X назви Y. Значення X, Y вводити з терміналу.
* перелік книг автора Z, що знаходяться в колекції;
* кількість книжок видання XX року, що є у бібліотеці.
 |
| 9 | 1. Створити файл, що містить відомості про наявність білетів і рейсів Аерофлоту. Структура запису: номер рейсу, пункт призначення, час виліту, час прибуття, кількість вільних місць у салоні. Кількість записів довільна.
2. Написати програму, яка виводить наступні дані:
	* час відправлення літаків у місто X;
	* наявність вільних місць на рейс у місто X з часом відправлення Y.

Значення X, Y вводяться по запиту з терміналу.  |
| 10 | 1. Створити файл, що містить відомості про асортимент взуття у магазині фірми. Структура запису: артикул, найменування, кількість, вартість однієї пари. Кількість записів довільна. Артикул починається з букви Д для дамського взуття, М — для чоловічого, П — для дитячого.
2. Написати програму, яка виводить наступні дані:
* про наявність та вартість взуття артикула X;
* асортиментний перелік дамського взуття з вказівкою найменування та числа пар кожної моделі, що є в наявності.
 |
| 11 | * 1. Створити два файли, що містять відомості про десять нападників команд «Динамо» та «Шахтар» відповідно: ім’я нападника, кількість забитих голів, зроблених гольових передач, кількість жовтих та червоних карток.
	2. Написати програму, що по даних, вилучених з цих файлів, створює новий третій файл, який містить ім’я, команду суму очок (голи + передачі) для 6 кращих гравців обидних команд. Імена та показники результативності футболістів вивести на екран
 |
| 12 | 1. Створити файл, що містить відомості про те, які з п’яти запропонованих дисциплін за вибором бажає слухати студент. Структура запису: прізвище студента, індекс групи, 5 дисциплін, середній бал успішності. Обрана дисципліна позначається «1», інакше — « ». Кількість записів — 25.
2. Написати програму, яка друкує список студентів, що бажають прослухати дисципліну X. Якщо кількість бажаючих перевищує 8 людей, то відібрати студентів, що мають вищій бал успішності.
 |
| 13 | 1. Створити файл, що містить відомості про відправлення поїздів дальнього сполучення із залізничного вокзалу. Структура запису: номер поїзду, станція призначення, час відправлення, час в дорозі, наявність білетів. Кількість записів довільна.
2. Написати програму, яка дозволяє отримати наступну довідкову інформацію:
	* час відправлення поїздів у місто X у часовому інтервалі від А до В годин;
	* наявність білетів на поїзд з номером XXX.
 |
| 14 | * + 1. Створити файл, що містить відомості про співробітників інституту. Структура запису: прізвище працівника, назва відділу, рік народження, стаж роботи, посада, заробітна плата. Кількість записів довільна.
		2. Написати програму, яка дозволяє потримати наступну інформацію:
* перелік співробітників пенсійного віку на сьогодні, з вказівкою стажу роботи;
* середній стаж працівників у відділі X.
 |
| 15 | 1. Створити файл, що містить відомості про пацієнтів клініки ока. Структура запису: прізвище пацієнта, стать, вік, місце проживання (місто), діагноз. Кількість записів довільна.
2. Написати програму, яка виводить інформацію:
* кількість іногородніх, які прибули в клініку;
* перелік пацієнтів старших X років з діагнозом Y. Значення X і Y ввести з терміналу.
 |

**Контрольні питання**

* 1. Пояснити, що означають наступні терміни: файл, запис, метод доступу, структура запису?
	2. Чи допустимі різні типи даних для елементів одного запису?
	3. Вказати, за допомогою яких операторів виконується запис даних в файл послідовного доступу, читання з файлу.
	4. Навести приклади використання файлів послідовного доступу.
	5. Як розпізнати кінець файлу даних? Як розпізнати файл на диску?

**ЛАБОРАТОРНА РОБОТА №8**

**Використання вкладених циклічних структур**

**Мета роботи**— оволодіти навичками алгоритмізації та програмування структур з вкладеними циклами.

**Теоретичні відомості**

Часто буває так, що при повтореннях змінюється не одна величина, а дві (чи навіть більше). І при кожному значенні однієї величини інша величина «пробігає» усі свої значення.

Цикл називається вкладеним, якщо він розміщується усередині іншого циклу. При першому проході, зовнішній цикл викликає внутрішній, який виконується до свого завершення, після чого управління передається в тіло зовнішнього циклу. При другому проході зовнішній цикл знову викликає внутрішній. І так до тих пір, поки не завершиться зовнішній цикл. Само собою, як зовнішній, так і внутрішній цикли можуть бути перервані командою break.

Вкладені цикли характеризуються рівнями вкладення. Зовнішній цикл має рівень вкладень 0, внутрішній - 1. Якщо ж тілом цього внутрішнього циклу знову є цикл, то його рівень вкладення буде 2 і т.д. У цьому випадку цикл з рівнем вкладення 1 є внутрішнім щодо циклу 2.

Параметри циклів у випадку вкладених циклів змінюються так: спочатку змінюється параметр внутрішнього циклу, набу­ваючи всіх своїх значень. Потім зовнішній цикл змінить зна­чення на один крок і знову параметр внутрішнього циклу набуде всіх значень. Так триває доти, доки параметр зовнішнього циклу не набуде всіх своїх значень.

Дуже часто вкладені цикли використовують для обчислення операцій з масивами, починаючи з двовимірного. Це дуже зручно, так як дозволяє перебирати всі поточні елементи за індексами цього масиву.

Приклад

Як приклад вкладених циклів виведемо на екран табличку множення від 1 до 10. Блок-схема такого алгоритму буде мати наступний вигляд:

i = 1; i <= 10; i=i+1

*i \* j*

початок

кінець

j = 1; j <= 10; j=j+1

 i=11

 j=11

**Код програми:**

#include <conio.h>

#include <stdio.h>

int main()

{

 clrscr();

 int i, j;

 for (i = 1; i <= 10; i++)

 {

 for (j = 1; j <= 10; j++)

 printf("%4d", i \* j);

 printf("\n");

 }

 return 0;

}

**Робоче завдання**

Обчислити на ЕОМ з заданою точністю ε значення екстремума на заданому відрізку [a;b], у відповідності до варіанта. На друк вивести таблицю значень функції f(x) тільки при „грубому” кроці, та знайдене значення екстремума і значення аргументу при якому він досягається, а також кількість зменшень кроку. Значення функцій подано у Таблиці 3.1.

Таблиця 3.1 — Варіанти завдань

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Варіант завдання | Вид функції *y*=*f(x)* | Вид екстремуму Хекс. | Діапазон зміни аргументу [a;b] | "Грубе" значення кроку h | Точність обрахунку екстремуму Е |
| 1 | 2+*x*-*x2* | Максимум 0,5 | [0;1,0] | 0,15 | 10-5 |
| 2 |  (1-x)4 | Мінімум 1,0 | [0,2;1,5] | 0,25 | 0,5\*10-4 |
| 3 | *cosx+chx* | Мінімум 0,0 | [-0,8;0,4] | 0,25 | 10-5 |
| 4 | *x1/3\*(1-x)2/3* | Максимум 0,33 | [0,1;0,6] | 0,1 | 10-5 |
| 5 |  x*3-6x2+9x+4* | Максимум 1,0 | [0,2;1,5] | 0,3 | 10-5 |
| 6 | x*3-6x2+9x+4*  | Мінімум 3,0 | [2;4] | 0,3 | 0,5\*10-5 |
| 7 | *2\*x2-x4* | Мінімум 0,0 | [-2;0,8] | 0,15 | 10-4 |
| 8 | *(x2-3x+2)/**(x2+2x-1)* | Мінімум 1,4 | [1;2] | 0,15 | 0,5\*10-4 |
| 9 |  *x\*(x-1)x/3* | Мінімум 0,75 | [0,1;1,2] | 0,2 | 10-5 |
| 10 |  *x\*e-x* | Максимум 1,0 | [0,1;1,5] | 0,25 | 10-5 |
| 11 | *ln2x/x*  | Максимум 7,389 | [6;8] | 0,15 | 10-5 |
| 12 | *x+1/x* | Мінімум 1,0 | [0,1;1,5] | 0,2 | 10-4 |
| 13 | *arctgx-ln(1+x2)/2*  | Максимум 1,0 | [0,15;1,5] | 0,2 | 10-5 |
| 14 |  │x│\*e-│x-1│ | Максимум -1,0 | [-2;-0,5] | 0,15 | 10-5 |
| 15 | ln2x/x  | Мінімум 1,0 | [0,1;1,9] | 0,2 | 10-4 |

**Контрольні запитання**

1. Вказати основні правила організації вкладених циклів.
2. Чи можливий вихід із внутрішнього циклу до його повного завершення?
3. Чому вихід із внутрішнього циклу при знаходженні екстремуму функції здійснюється до досягнення правої границі інтервалу [a;b]?
4. Вказати, які можливості має досліджувана мова програмування для побудови ітераційних циклічних структур із заданим числом повторів.