МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра АТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

«ІНФОРМАТИКА»

(Частина 2)

для студентів напряму 6.050903 “Телекомунікації”

денної і заочної форм навчання.

Розглянуто на засіданні кафедри
«Автоматика і телекомунікації»
протокол № 9 від 30.08.2010р.

Затверджені на засіданні
Навчально-видавничої ради ДонНТУ
Протокол № 4 від 07.10.2010р. р.№344

ДОНЕЦЬК – 2010

Методичні вказівки до виконання лабораторних робіт з курсу «Інформатика» (частина 2) для студентів напряму 6.050903 “Телекомунікації” денної і заочної форм навчання. /Яремко І.М., Долгіх І.П. – Донецьк, ДонНТУ, 2010. - 50с.

Коротко розглянуто теоретичний матеріал, наведені приклади. Подані варіанти завдань до виконання лабораторних робіт.

Укладачі: ст. викладач Яремко І.М.,

ст. викладач Долгіх І.П.

Відповідальний за випуск:
Зав. кафедрою «Автоматика і телекомунікації»
к.т.н., доцент Бессараб В.І.

Рецензент: к.т.н., доцент кафедри «Автоматизовані системи управління» В.А. Світлична

ЛАБОРАТОРНА РОБОТА №1
ПОБУДОВА ПРОГРАМ РОЗГАЛУЖЕНОЇ СТРУКТУРИ

Ціль роботи: здобуття практичних навичок запису арифметичних виразів і використання в програмі оператора умови, умовної операції й оператора перемикача.

1. Методичні вказівки до виконання роботи

У програмі мовою Сі у виразах бажано використовувати константи й змінні одного типу. Якщо відбувається змішування типів, то компілятор не вважає програму неправильною, а використовує набір правил для автоматичного перетворення типів:

1. Якщо операція виконується над даними двох різних типів, обидві величини приводяться до "вищого" із двох типів. Цей процес називається "підвищенням" типу.

2. Послідовність імен типів, упорядкованих від "вищого" до "нижчого", виглядає так:

double float long int short char

Застосування ключового слова unsigned підвищує ранг відповідного типу даних зі знаком.

3. В операторі присвоювання кінцевий результат обчислення виразу в правій частині приводиться до типу змінної, якій повинне бути привласнене це значення. Даний процес може привести й до "підвищення" і до "зниження" типу. Для збереження точності обчислень при арифметичних операціях усі величини типу ***float*** перетворюються в дані типу ***double***, а типи ***char***і***short*** перетворюються до типу ***int.*** Це суттєво зменшує помилку округлення.

Для організації програм розгалуженої структури мовою Сі використовується умовний оператор, умовна операція й оператор перемикач.

Умовний оператор

Умовний оператор має дві форми запису:

If (вираз) оператор1;

та

if (вираз) оператор1;else оператор2;

Якщо вираз дійсний, то виконується оператор1, якщо він хибний, то при використанні форми 1 управління передається наступному оператору, а при застосуванні другої форми запису – виконується оператор2.

Якщо необхідно виконати кілька дій, то використовується складений оператор {}.

Приклад. Розглянемо фрагмент програми

```
if (i<j) i++;
else { j=i-3;  i++; }
```

Якщо значення *i* більше, ніж *j*, то відбувається збільшення його на 1.

Якщо значення *j* більше, ніж *i*, то виконуються дві дії: присвоєння нового значення змінній *j* і збільшення *i*. У цьому випадку в галузі else використовується складений оператор для об'єднання двох дій.

Допускається використання вкладених операторів if. Оператор може бути вкладений у фазу if або else іншого оператора if. Якщо немає фігурних дужок, то ключове слово else ставиться відноситься до найближчого оператора if, у якого немає гілки else.

Приклад. Розглянемо два фрагменти програми

```
if (a==b)
  { if (a==0) b=2;
   }
else a=2;

if (a==b)
if (a==0) b=2;
else a=2;
```

У першій програмі else відноситься до першого оператора if, а в другій – до другого оператора if.

Для запису умовного виразу можуть використовуватися наступні операції порівняння й логічні операції:

== рівно != не рівно

<,<= менше, менше або рівно >,>= більше, більше або рівно

! інверсія && логічне “І”

|| логічне “АБО”.

Результатом порівняння є дане типу int значення, що приймає значення 0 при невиконанні умови порівняння (хибність) і значення 1 при виконанні умови порівняння (істина).

Умовна операція

Це короткий спосіб запису оператора if. Форма запису оператора наступна:

вираз1 ? вираз2 : вираз3;

"Вираз1" повинен бути цілого або плаваючого типу або вказівник. Якщо "вираз1" дорівнює нулю (хибність), то обчислюється "вираз3", і його значення є результатом операції. Якщо значення "виразу1" відмінне від нуля (істинно), то результатом операції є значення "виразу2".

Приклад. Знаходження максимального із двох значень змінних a й b, і збереження його в змінній max.

max=(a<b) ? b : a;

Умовну операцію зручно використовувати в тих випадках, коли змінній необхідно присвоїти одне з двох можливих значень.

Оператор – перемикач

Оператор, призначений для організації вибору одного з множини варіантів. Загальний вид оператора перемикача:

switch(вираз) {
case мітка1:оператори1;
case мітка2:оператори2;
…
case мітка n: оператори n;
default: оператори;
}

Значення "виразу" обчислюється й порівнюється з "мітками" (зазвичай, це цілі або символьні константи). У випадку збігу виконується група операторів, відповідна до мітки. Оператор default виконується, якщо жоден з попередніх операторів не виконався. Можливе використання декількох міток перед групою операторів. Наявність гілки default необов'язково. Бажане наприкінці групи операторів, відповідних до кожної мітки, використовувати оператор break для завершення виконання оператора перемикача.

Приклад. Виконання арифметичної операції по заданому знаку у змінній sign. Фрагмент програми буде наступним:

switch(sign){
case '-':x=y-z; break;
case'+':x=y+z; break;
case '*':x=y*z; break;

case '/':x=y/z; break;
default:printf("Невідома операція\n"); }
printf ("'Результатоперації=%f\n" ,x);

Оператор перемикач може бути вкладено один в іншій, при цьому їх мітки можуть збігатися.

Оператор розриву

Форма запису оператора розриву: **break;**

Використовується в операторах циклу й в операторі switch. Його виконання приводить до виходу із зазначених конструкцій і перехід до наступного оператора програми. Якщо оператор розриву перебуває усередині деякої сукупності вкладених структур, його дія поширюється тільки на саму внутрішню структуру, у якій він безпосередньо міститься.

Приклад. Програма, яка за введеними значеннями довжин боків з'ясовує, чи можна скласти з них трикутник і обчислює його площу. При обчисленні значення змінної s використовується операція приведення типу: (float). Вона необхідна для збереження дробової частини, тому що тип вихідних змінних int, а тип результату float.

```
#include<stdio .h>
 #include<math .h>
void main()
{ int a=3,b=5,c=7;
float s,p;
if (a>=(b+c)||b>=(a+c)||c>=(a+b))
printf ("трикутник не складається\n");
else {
     s=(float)l/2*(a+b+c);
p=sqrt(s*(s-a)*(s-b)*(s-c));
printf( "p=%f\n'1 ,p);
    }
}
```

1. Контрольні питання

1. Правила для автоматичного перетворення типів у виразі.
2. Роль ключового слова unsigned при заданні типу.
3. Використання в програмі умовної операції.
4. Роль оператора розриву в операторі – перемикачу.
5. Форми запису оператора умови.

3. Завдання

Скласти блок-схему алгоритма і програму мовою Сі/Сі++ згідно варіанта (табл.1.1).

Таблиця 1.1. Варіанти завдань

№	**функціязавдання y(x)=…**
1.	$\begin{cases} \lg(x) & \text{при} \quad 4 <= x < 6, \\ \sin^2(x) & \text{при} \quad x >= 6, \\ e^{0.1x} & \text{при} \quad -3 <= x < 4, \\ \dfrac{\cos(x)}{x+10} & \text{при} \quad x < -3 \end{cases}$
2.	$\begin{cases} e^{\sqrt{x}} & \text{при} \quad x > 2, \\ \cos^2(x) & \text{при} \quad 2 >= x > 1, \\ \lg(20x) & \text{при} \quad 1 >= x > 0.5, \\ e^{0.2x} & \text{при} \quad x <= 0.5 \end{cases}$
3.	$\begin{cases} \dfrac{1}{x} & \text{при} \quad x > 4, \\ e^{-x} & \text{при} \quad 2 < x <= 4, \\ x^2 & \text{при} \quad -1 < x <= 2, \\ \dfrac{\lvert x-1 \rvert}{2x} & \text{при} \quad x <= -1 \end{cases}$
4.	$\begin{cases} \dfrac{2}{x} & \text{при} \quad x < -5, \\ x\sin x & \text{при} \quad 2 > x >= -5, \\ \dfrac{x+10}{2+x^2} & \text{при} \quad 3 > x >= 2, \\ x + \sqrt[3]{x} & \text{при} \quad x >= 3 \end{cases}$
5.	$\begin{cases} \sqrt{x} & \text{при} \quad x > 7, \\ e^{-x} & \text{при} \quad 2 < x <= 7, \\ e^{x} & \text{при} \quad -1 < x <= 2, \\ \sin(x) & \text{при} \quad x <= -1 \end{cases}$

6.	$\begin{cases} 3x^3 & \text{при} \quad x > 20, \\ \ln(x+3) & \text{при} \quad 20 >= x > 10, \\ e^{-x^2} & \text{при} \quad 10 >= x > 3, \\ \lvert \sin(x) \rvert & \text{при} \quad 3 >= x \end{cases}$
7.	$\begin{cases} \frac{\sin^2(x)}{2} & \text{при} \quad x > 20, \\ \sqrt[3]{x} & \text{при} \quad 20 >= x >= 6, \\ \sin^2(x) & \text{при} \quad 6 >= x > -4, \\ 0 & \text{при} \quad x <= -4 \end{cases}$
8.	$\begin{cases} \sin(\sqrt{x}) & \text{при} \quad x > 30, \\ \sqrt{x} & \text{при} \quad 30 >= x > 10, \\ \lg(0.5 * x) & \text{при} \quad 10 >= x > 2, \\ \lvert \sin(x) \rvert & \text{при} \quad x <= 2 \end{cases}$
9.	$\begin{cases} 2\sqrt{(x^2+15)} & \text{при} \quad x < -6, \\ 4\cos(x) & \text{при} \quad -6 <= x < 2, \\ \frac{\sin(x-3)}{2} & \text{при} \quad 2 <= x < 10, \\ \frac{tgx}{10} & \text{при} \quad x >= 10 \end{cases}$
10.	$\begin{cases} 1+x^2 & \text{при} \quad x > 25, \\ 2+x^2 & \text{при} \quad 25 >= x > 8, \\ 3+x^2 & \text{при} \quad 8 >= x > 2, \\ 4+x^2 & \text{при} \quad x <= 2 \end{cases}$
11.	$\begin{cases} x \lvert x+21 \rvert & \text{при} \quad x < -14, \\ x^2 \ln \lvert x^2+48 \rvert & \text{при} \quad -14 <= x < -5, \\ \frac{x}{3} + \sqrt{(x^2+16)} & \text{при} \quad -5 <= x < 0, \\ 2 + \frac{x}{3} & \text{при} \quad x >= 0 \end{cases}$
12.	$\begin{cases} x + (\lvert x \rvert - 12)^2 & \text{при} \quad x < -8, \\ \frac{\sin(x)+3}{\cos(x)+15} & \text{при} \quad -8 <= x < 7, \\ x\sqrt{(x^2-36)} & \text{при} \quad 7 <= x < 10, \\ x+8 & \text{при} \quad x >= 10 \end{cases}$

13.
$$\begin{cases} \dfrac{\cos(x)+14}{\sin(x)+7} & при \quad x <= 4, \\ \sqrt[3]{(x+\ln(|x-8|+10))} & при \quad 4 < x < 12, \\ \sqrt{(x-13)} & при \quad 12 <= x < 38, \\ 5x & при \quad x >= 38 \end{cases}$$

14.
$$\begin{cases} \sqrt{|x|}+\dfrac{\sin(x)}{2} & при \quad x < -12, \\ \sin(x-5) & при \quad -12 <= x < 8, \\ \dfrac{\sqrt{(x+6)}}{x+3} & при \quad 8 <= x < 10, \\ \dfrac{4}{3+x} & при \quad x >= 10 \end{cases}$$

15.
$$\begin{cases} \dfrac{x\sin(x)+5}{2+\sin(x)} & при \quad x < 5, \\ \sqrt{|x-7|}+\dfrac{x}{2} & при \quad 5 <= x < 10, \\ \lg(x) & при \quad 10 <= x < 111, \\ 2x & при \quad x >= 111 \end{cases}$$

16.
$$\begin{cases} \sqrt{(|x|-1)} & при \quad x <= -20, \\ \dfrac{x+\sin(x)}{2} & при \quad -20 < x < 6, \\ x^2+\ln(x-2) & при \quad 6 <= x < 12, \\ x^2+\ln(10) & при \quad x >= 12 \end{cases}$$

17.
$$\begin{cases} \lg(1.5x) & при \quad 4 <= x < 7, \\ \sin^4(x) & при \quad x >= 7, \\ 0.5e^{0.1x} & при \quad -3.5 <= x < 4, \\ \dfrac{\cos(x)}{|x|+10} & при \quad x < -3.5 \end{cases}$$

18.
$$\begin{cases} \dfrac{\sin(x)+2}{\cos(x)+10} & при \quad x <= 5, \\ \sqrt[3]{(2+\ln(|x-8|+2))} & при \quad 5 < x < 15, \\ \sqrt{x-13} & при \quad 15 <= x < 38, \\ 5+x & при \quad x >= 38 \end{cases}$$

19.	$\begin{cases} \frac{x}{2} & \text{при } x > 5, \\ 10e^{-x} & \text{при } 2 < x <= 5, \\ \frac{3+x^2}{2x} & \text{при } -2 < x <= 2, \\ \frac{\lvert x-100\rvert}{2x} & \text{при } x <= -2 \end{cases}$
20.	$\begin{cases} \sqrt{x} & \text{при } x > 9, \\ 5e^{-x} & \text{при } 2 < x <= 9, \\ e^{x} & \text{при } 1 < x <= 2, \\ \cos(x) & \text{при } x <= 1 \end{cases}$
21.	$\begin{cases} x^3 & \text{при } x > 10, \\ 2x & \text{при } 10 >= x > 3, \\ 0{,}5x & \text{при } 3 >= x > -5, \\ \sqrt[3]{\lvert x+2\rvert} & \text{при } x <= -5 \end{cases}$
22.	$\begin{cases} x^2 & \text{при } x > 10, \\ \lg(x) & \text{при } 10 >= x > 1, \\ \sqrt{\lvert x-15\rvert} & \text{при } 1 >= x >= -1, \\ 4 & \text{при } x < -1 \end{cases}$
23.	$\begin{cases} e^{-x} & \text{при } x <= 0, \\ \sqrt[3]{x+1} & \text{при } 0 < x <= 7, \\ \frac{x+13}{10} & \text{при } 7 < x <= 87, \\ 10(x-86) & \text{при } x > 87 \end{cases}$
24.	$\begin{cases} \frac{x+75}{\sqrt{x}+5} & \text{при } x >= 25, \\ 0{,}4x & \text{при } 25 > x >= 5, \\ \sqrt{x-1} & \text{при } 5 > x >= 2, \\ e^{x-2} & \text{при } x < 2 \end{cases}$
25.	$\begin{cases} \sqrt{x-1} & \text{при } x >= 5, \\ \frac{x+15}{x+5} & \text{при } 5 > x >= 0, \\ 3\cos x & \text{при } 0 > x >= -2\pi, \\ 3 & \text{при } x < -2\pi \end{cases}$

26.	$\begin{cases} \sqrt{x^2+16} & \text{при } x >= 3, \\ x & \text{при } 3 > x >= 1, \\ x^2+1 & \text{при } 1 > x >= 0, \\ e^{-x} & \text{при } x < 0 \end{cases}$
27.	$\begin{cases} 11 & \text{при } x > 5, \\ 2x+1 & \text{при } 5 >= x > 2, \\ x^2+1 & \text{при } 2 >= x > 0, \\ e^{-x} & \text{при } x <= 0 \end{cases}$
28.	$\begin{cases} 0 & \text{при } x >= \pi \\ \sin x & \text{при } \pi > x >= 0, \\ 0 & \text{при } 0 > x >= -2, \\ (x+2)^2 & \text{при } x < -2 \end{cases}$
29.	$\begin{cases} 4+(x-2)^2 & \text{при } x > 2, \\ 2^x & \text{при } 2 >= x > 1, \\ x+1 & \text{при } 1 >= x > 0, \\ e^x & \text{при } x <= 0 \end{cases}$
30.	$\begin{cases} 27+(x-3)^3 & \text{при } x > 3, \\ x^3 & \text{при } 3 >= x > 1, \\ x & \text{при } 1 >= x > 0, \\ \frac{\sin^2 x}{2} & \text{при } x <= 0 \end{cases}$

Лабораторна робота №2

ПРОГРАМУВАННЯ ЦИКЛІВ

Ціль роботи: вивчення операторів циклу й особливостей їх застосування.

1. Методичні вказівки до виконання роботи

Оператор циклу while

Даний оператор циклу має вигляд:

while **(вираз) оператор;**

Спочатку обчислюється значення виразу, якщо воно хибне, то управління передається до наступного за циклом оператору. Якщо умова істинна, то виконується тіло оператора. Оператор може бути порожнім, простим і складовим. Порожній оператор складається тільки з ";". При виконанні оператора нічого не відбувається. Використовується, коли тіла циклу не потрібно, хоча по синтаксису потрібний хоча б один оператор.

Приклад. Розглянемо наступний фрагмент програми

index=1; while(index++<=5) printf("Привіт!\n");

Змінна *index* відповідає за число повторень циклу. У результаті роботи наведеного фрагмента програми на екран п'ять разів буде виведене слово "Привіт!".

Оператор циклу for

Даний оператор циклу має вигляд:

For(**вираз** 1; **вираз**2; **вираз**3) **оператор;**

"Вираз1" задає початкові умови. Він виконується всього один раз на початку циклу. "Вираз2" задає перевірку умови. Перевірка проводиться перед кожним можливим виконанням циклу. Коли "вираз2" стає хибним, цикл закінчується. "Вираз3" обчислюється наприкінці виконання кожного тіла циклу, модифікує вираз, що перевіряється.

"Оператор" може бути простим або складеним.

Приклад. Скласти програму обчислення квадратів цілих чисел від 0 до 9.

voidmain()

```
{int i;
for(i=0;i<10;i++)
printf("квадрат числа %d = %d\n", i, i*i);
}
```

Можливості оператора циклу for:

1. Можна лічити в порядку убування й зростання значень параметра циклу.

2. Крок зміни параметра циклу може бути будь-яким.

Наприклад, *for(n=2;n<60;n+=13)* оператор;

3. Можна вести підрахунок за допомогою символів.

Наприклад, *for (charch=,a';ch<='z,;ch++)* оператор;

Цей оператор працює, оскільки символи в пам'яті машини розміщаються у вигляді чисел.

4. У якості "виразуЗ" можна використовувати будь-який істинний вираз.

Його значення буде змінюватися при кожній ітерації.

Приклад. Фрагмент програми для табулювання функції y=5x+10 може виглядати так

```
for(x=l;y<=75;y=5*x++ + 10) printf(" %d %d\n",x,y);
```

У якості "виразуЗ" у даному прикладі використовується формула обчислення заданої функції. Зміна параметра x задається в самій формулі за допомогою постфіксної операції інкременту.

5. Можна опустити один або більше виразів. Але при цьому не опускаються символи ";". Необхідно тільки включити в тіло циклу кілька операторів, які, зрештою , приведуть до завершення його роботи.

Приклад. Розглянемо фрагмент програми

```
ans=2;
for(n=3;ans<=25;) ans=ans*n;
```

У цьому випадку пропущений"вираз3". Зміна параметра циклу відбувається в тілі циклу. Ці ж дії можна було б записати, використовуючи в якості тіла циклу, порожній оператор.

```
ans=2;
for(n=3;ans<=25;ans*=n);
```

Якщо опустити всі три вирази в операторі циклу for, то можна задати нескінченний цикл, оскільки порожня умова завжди вважається дійсною.

Наприклад, for(;;) { оператори}

6. Перший вираз не обов'язково повинен ініціювати змінну. Необхідно тільки пам'ятати, що "вираз1" виконується тільки один раз перед тем, як інші частини циклу почнуть виконуватися.

Наприклад,наступний цикл буде працювати, поки не буде введене число більше за 10.

for(printf("введіть числа\n");num<=10;scanf("%d",&num);

7. Параметри, що входять у вирази можна змінювати в тілі циклу.

8. Операція "кома" дозволяє включати в специфікацію циклу деклька ініціалізуючих і корегувальних виразів. Операція "кома" (,) зв'язує два вирази в один , причому лівий вираз буде виконуватися першим.

Наприклад, for(i=0,j=10;i<j;i++,j--) оператор;

Оператор do

Оператор має наступну форму запису:

do

оператор;

*while(**вираз**);*

Оператор do використовується в тих випадках, коли тіло циклу повинне виконатися хоча б один раз. Спочатку виконується оператор, потім обчислюється вираз. Якщо вираз дійсний (true), то цикл повторюється. Якщо вираз помилковий (false), то управління передається наступному за циклом оператору.

Приклад. Реалізація фрагмента програми видачі підказки з використанням оператора do.

```
do {
printf("введіть Y або N\n");
scanf("%c",&c);
}
while(c!='Y' && c!= 'N')
```

Доти поки не буде введено один із правильних відповідей, оператор буде видавати підказку та зчитувати введений символ.

Оператор продовження

Оператор має такий вигляд: continue;

Його дія полягає в перериванні виконання тіла циклу. Після цього викликається наступна ітерація (крок) цього циклу.

Приклад. Фрагмент програми виведення парних чисел до 100.

```
for(i=0;i< 100;i++)
{ if(i%2) continue;
printf("%d\n",i);}
```

Якщо значення *i* - парне, то залишок від ділення на 2 дорівнює нулю й, отже, результат перевірки умови в операторі if - хибність, і буде виконаний оператор друку. А якщо ні, то, результат перевірки умови - істина й, отже, буде виконаний оператор продовження. У цьому випадку ми, минаючи оператор друку відразу, переходимо до наступної ітерації (наступному значенню *i*).

В операторах циклу дуже зручно використовувати оператор розриву для переходу до наступного оператора програми.

Приклад. Фрагмент програми виходу з нескінченного циклу при введенні числа, відмінного від нуля.

```
for(;;){
scanf("%d",&num);
if(!num) break;}
```

2. Контрольні питання

1. Що таке цикл?
2. Які оператори циклу мови Сі/Сі++ вам відомі?
3. Особливості застосування оператора for.
4. До якого типу належить цикл *for* (з попереднім- або постумовою)?
5. У чому відмінність циклу *for* у мовах Паскаль і Сі?
6. Що означають оператори "++", "--"?
7. Використання оператора-продовження й оператора-розриву в циклах.

3. Завдання

Скласти програму для обчислення виразу з використанням циклічних операторів, враховуючи, що x – змінювана величина (табл.2.1). Представити варіанти програми з різними типами циклів.

При виведенні результатів взяти до уваги, що їх кількість може бути більшою від кількості рядків на екрані.

ЛАБОРАТОРНА РОБОТА №3
ОБРОБКА МАСИВІВ ДАНИХ

Ціль роботи: ознайомитися з даними типу масив і основними прийомами програмування задач обробки масивів.

1. Методичні вказівки до виконання роботи

Масив - це група елементів однакового типу. Форма оголошення масиву мовою Сі наступна:

тип_даних ім'я_масиву[розмір];

"Тип_даних" задає тип елементів масиву. "Розмір" - кількість елементів у ньому. Елементи масиву в Сі нумеруються з нуля!

Приклад. Опис одномірного масиву з 10 цілих чисел і двовимірного масиву дійсних чисел з 10 рядків і 10 стовпців колонок,шпальт.

int page[9]; float bigmas[9][9];

Звертання до елементів цих масивів відбувається походить в такий спосіб:

page [1]- звертання до другого елемента масиву,

bigmas[0][0] - звертання до елемента двовимірного масиву.

При завданні масиву можлива і його ініціалізація. У цьому випадку значення, що привласнюються >, вказуються у квадратних дужках .

Приклад. Ініціалізація одномірного масиву цілих чисел:

int s[2]={ 1,2,3};

Якщо розмір масиву не зазначений, то він визначається по числу початкових значень. Але рекомендується завжди вказувати розмір масиву, що оголошується.

int day[]= {31,28,31,30,31,30,31,31,30,31,30,31};

При ініціалізації багатомірних масивів початкові значення для кожного нового рядка містяться у фігурні дужки. Якщо окремих фігурних дужок нема, то ініціалізація проводиться в мірі зростання індексів.

Приклади ініціалізації двовимірного масиву:

int s[l][2]={{4,5,6},{7,8,9}};

int f[2][3]={ 10,11,12,13,14};

char p[2] [2]={{'n'}, {'y'}};

Масив s ініциалізується повністю заданими значеннями. У масиві f з його шести значень (розмір масиву f - 2 рядки й 3 стовпці) ініціалізується тільки перші 5 елементів (це елементи з індексами 0,0 0,1 0,2 1,0 1,1). У масиві p ініциалізується тільки 2 елемента: p[0][0]= 'n' і p[1][0]= 'y'.

Якщо не проініціалізувати елементи масиву перед початком роботи з ним, то зовнішні й статичні масиви ініціалізуються нулем, а автоматичні й реєстрові будуть містити "сміття", що залишилося в цій ділянці пам'яті.

Якщо заданий розмір масиву, то значення, не задані явно, визначаються залежно від класу пам'яті.

Використання вказівників при роботі з масивами

У мові Сі існує сильний взаємозв'язок між вказівниками й масивами, настільки сильна, що вказівники й масиви фактично треба розглядати одночасно.

Будь-яка дія, яка досягається індексуванням масиву, може бути виконане й за допомогою вказівника. Варіант із вказівником у загалі–то буде швидший, але він, принаймні для початківців, дещо важкий для розуміння.

Опис int a[10] визначає масив a розміром в 10 елементів, тобто це блок з 10 послідовних об'єктів, іменованих a[0], a[1],..., a[9]. Запис a[i] позначає елемент в i- тій позиції від початку. Якщо РФ - це вказівник на ціле значення, описаний як

int * pa;

то присвоювання: pa=&a[0] встановлює в pa посилання на нульовий елемент масиву a, тобто рамістить адресу a[0]. Тепер присвоювання x=*pa копіює вміст a[0] у x.

Якщо вміст *pa* вказує на окремий елемент масиву *a*, то по визначенню *pa+1* вказує на наступний елемент, і, взагалі, (*pa– i*) вказує на i- тий елемент перед pa, а (*pa+i*)- на i- тий елемент після. Таким чином, якщо *pa* вказує на *a[0],* то*(*pa+1)*відноситься до вмісту *a[1]; pa+i*є адреса *a[i]*, а *(*pa+i)* є вміст *a[i]*.

Ці зауваження справедливі незалежно від типу змінних у масиві *a*. Визначення операції "додавання 1 до посилання" і іншої посилальної арифметики має на увазі масштабування, пов'язане з розміром пам'яті для об'єкта, на який вказує посилання. Таким чином, в *pa+i* значення *i*, перш ніж буде додано до *pa*, буде помножено на розмір об'єкта, на який вказує *pa*. Очевидно, що між індексуванням і посилальною арифметикою зв'язок дуже тісний. Фактично, будь-яке згадування масиву приводиться транслятором до посилання на початок цього масиву, тобто ім'я масиву є посилальне вираження. Це приводить до невеликого числа корисних наслідків. Оскільки ім'я масиву є синонім до місця розташування нульового елемента, то присвоювання

pa=&a[0]

можна записати й у такому вигляді:

pa=a.

Не дивно тепер, принаймні на перший погляд, що значення *a[i]* можна записати як **(a+i)*. Обчислюючи *a[i]*, транслятор відразу ж переводить його в **(a+i)*; ці дві форми повністю еквівалентні. Застосовуючи операцію & до обох частин цієї рівності, одержуємо, що *&a[i]*і*a+i* також ідентичні: *a+i* - адреса *i*-го елемента відносно *a*. З іншого боку, якщо *pa*– посилання, то її можна використовувати з індексом: *pa[i]* ідентично **(pa+i)*. Коротше, будь-який масив і індексне вираження можна записати як посилання й зміщення і, навпаки, причому це можна робити навіть в одному операторі.

Однак між ім'ям масиву й вказівником є одна відмінність про яку слід завжди пам'ятати. Вказівник є змінна, так що *pa=a* й *pa++* суть осмислені операції. Ім'я ж масиву - константа, а не змінна, тому конструкції начебто *a=pa* або *a++*, або *p=&a* неприпустимі.

Приклад. Програма роздруківки вмісту одномірного масиву з використанням вказівника. Масив попередньо проініціалізовано.

```
#include<stdio.h>
voidmain() {
int p[5]={l,2,3,4,5};
int *ref;
ref=p;
printf("\n");
for(int i=0;i<5;i++)
printf(" %d\t", *(ref+i));}
```

Приклад. Фрагмент програми, що реалізує заповнення двовимірного масиву випадковими елементами з використанням вказівника.

```
int a[5][6],pa;
……..…
randomize();
pa=&a[0][0];
for(i=0;i<5*6;i++)
*(pa+i)=random(2);
```

Розглянемо тепер, як одержати доступ до елемента багатомірного масиву, використовуючи вказівник. Припустимо, у програмі описаний тривимірний масив і вказівник на нього:

```
int arr[L][M][K], *ptr;
ptr=&arr [0] [0] [0];
```

Масив arr складається з L елементів, кожний з яких – двовимірний масив M на N. Кожний масив M на N у пам'яті розташовується по рядках.

Необхідно одержати доступ до елемента arr[i][j[k]. Послідовно це обчислюється так:

ptr– адреса 0- го масиву M на N

ptr+i*(M*N) – адреса i-го масиву M на N

ptr+i*(M*N)+j*N – адреса j-го рядка i-го масиву M на N

ptr+i*(M*N)+i*N+k– адреса елемента arr[i][j][k]

(ptr+i(M*N)+i*N+k) – значення елемента arr[i][j][k].

2. Контрольні питання

1. Які способи опису масивів застосовуються в мові *Ci*?
2. Ініціалізація масиву при його описі .
3. Опис вказівника на масив і його ініціалізація.
4. Звертання до елемента масиву за допомогою вказівника.
5. Одержання адреси й значення елемента багатомірного масиву.

3. Завдання

Розробити блок-схему алгоритму й програму мовою C/C++ по обробці одномірного масиву.

Примітка. Розмірності масивів задаються іменованими константами.

Частина 1. Обробка одномірного масиву

Розробити блок-схему алгоритму й програму мовою C/C++ по обробці одномірного масиву.

Примітка. Розмірності масивів задаються іменованими константами.

Варіант 1. Даний одномірний масив A, що складається з N елементів. Скільки значень елементів у масиві A зустрічається більш одного разу?

Варіант 2. Даний одномірний масив A, що складається з N елементів. Скільки значень елементів зустрічається в масиві по 3 рази?

Варіант 3. Даний одномірний масив A, що складається з N елементів. Переписати в одномірний масив B всі елементи, розташовані між максимальним і мінімальним значеннями.

Варіант 4. Даний одномірний масив A, що складається з N елементів. Визначити кількість чисел, що входять у масив по одному разу.

Варіант 5. Даний одномірний масив A, що складається з N елементів. Підрахувати максимальну кількість нулів, що йдуть підряд.

Варіант 6. Даний одномірний масив А, що складається з N елементів. Перенести в початок масиву всі додатні елементи, а в кінець масиву - всі від'ємні.

Варіант 7. Даний одномірний масив А, що складається з N елементів. Перенести в початок масиву всі парні елементи, а в кінець масиву - усі непарні.

Варіант 8. Даний одномірний масив А, що складається з N елементів. Виключити з масиву перший додатний елемент, що іде за максимальним.

Варіант 9. Даний одномірний масив А, що складається з N елементів. Виключити з масиву всі нульові елементи, розташовані між максимальним і мінімальним елементами.

Варіант 10. Даний одномірний масив А, що складається з N елементів. Виключити з масиву перший додатний елемент, що передує максимуму .

Варіант 11. Даний одномірний масив А, що складається з N елементів. Підрахувати максимальну кількість від'ємних елементів, що йдуть підряд.

Варіант 12. Даний одномірний масив А, що складається з N елементів. Знайти перший і останній додатні елементи масиву й підрахувати кількість елементів, розташованих між ними.

Варіант 13. Даний одномірний масив А, що складається з N елементів. Підрахувати максимальну кількість додатних елементів, розташованих між нулями.

Варіант 14. Даний одномірний масив А, що складається з N елементів. Вважаємо, що від'ємні елементи розбивають його на групи. Знайти групу додатних елементів масиву з максимальною сумою.

Варіант 15. Даний одномірний масив А, що складається з N елементів. Вважаємо, що від'ємні елементи розбивають його на групи. Знайти кількість отриманих груп, що містять нулі.

Варіант 16. Даний масив чисел $a_1,...,a_N$. З'ясувати, чи є в даному масиві два додатних елементи , що йдуть ідуться підряд . Підрахувати кількість таких пар.

Варіант 17. Даний масив цілих чисел $a_1,...,a_N$. Підрахувати кількість пар елементів, що задовольняють умові $a_i < a_{i+1}$.

Варіант 18. Даний масив цілих чисел $a_1,...,a_N$. Знайти в даній послідовності всі пари a_i, a_{i+1},такі, що $a_i = 0$ та a_{i+1} кратне 10.

Варіант 19. Даний масив чисел $a_1,...,a_N$. З'ясувати, чи є в даному масиві два від'ємні елементи, що йдуть підряд. Підрахувати кількість таких пар.

Варіант 20. Даний масив цілих чисел $a_1,...,a_N$. Підрахувати кількість пар елементів, що задовольняють умові $a_i > a_{i+1}$.

Варіант 21. Даний одномірний масив А, що складаєтьсяз N елементів. Підрахувати максимальну кількість додатних чисел, що йдуть підряд .

Варіант 22. Даний одномірний масив А, що складається з N елементів. Підрахувати максимальну кількість від'ємних чисел, що йдуть підряд.

Варіант 23. Даний одномірний масив А, що складається з N елементів. Виключити з масиву всі додатні елементи, розташовані між максимальним і мінімальним елементами.

Варіант 24. Даний одномірний масив А, що складається з N елементів. Виключити з масиву всі від'ємні елементи, розташовані між максимальним і мінімальним елементами.

Варіант 25. Даний одномірний масив А, що складається з N елементів. Виключити з масиву перший від'ємний елемент, що прямує за максимальним.

Частина 2. Обробка двовимірного масиву

Розробити блок-схему алгоритму й програму мовою С/С++ по обробці масиву.

Примітка. Визначити масив як динамічний. Оформити обробку функцією з передачею вказівника на масив.

Варіант 1. Дана матриця цілих чисел. У рядках, усі елементи яких парні, розташувати елементи у зворотному порядку.

Варіант 2. Дана матриця символів. Підрахувати кількість рядків, у яких літер більше, ніж цифр.

Варіант 3. Дана матриця цілих чисел. Зібрати всі додатні елементи масиву вище головної діагоналі (заповнення здійснювати по рядках).

Варіант 4. Дана матриця цілих чисел. Зібрати всі нульові елементи вище головної діагоналі (заповнення здійснювати паралельно головній діагоналі).

Варіант 5. Дана матриця символів. Написати програму звертання до кожного елемента цієї матриці, якщо вважати, що імена рядків - літери алфавіту (по зростанню), а імена стовпців - цілі числа (по зростанню).

Варіант 6. Дана матриця дійсних чисел. Знайти максимальний елемент і найближчий до нього (за значенням) елемент матриці. Пошук здійснювати у квадратному контурі , центром якого є максимум, а довжина боку - п'ять елементів масиву.

Варіант 7. Дана матриця дійсних чисел. Знайти мінімальний елемент і найближчий до нього (за значенням) елемент матриці. Пошук здійснювати в напрямку,паралельному головній діагоналі.

Варіант 8. Дана матриця дійсних чисел. Знайти число з максимальною дробовою частиною, переставити рядки й стовпці так, щоб це число стояло в лівому верхньому куті .

Варіант 9. Дана матриця цілих чисел. Підрахувати кількість елементів, що передують максимуму, і кількість елементів, що прямують за мінімумом.

Варіант 10. Дана матриця символів. Визначити рядок, у якому максимальна кількість літер.

Варіант 11. Дана матриця цілих чисел. Зібрати всі від'ємні елементи вище побічної діагоналі (заповнення здійснювати по рядках).

Варіант 12. Дана матриця дійсних чисел. Знайти максимальний елемент і найбільш віддалений від нього (за значенням) елемент матриці. Пошук здійснювати у квадратному контурі , центром якого є являється максимум, а довжина боку - три елементи масиву.

Варіант 13. Дана матриця дійсних чисел. Знайти мінімальний елемент і найбільш віддалений від нього (за значенням) елемент матриці. Пошук здійснювати в напрямку, паралельному головній діагоналі.

Варіант 14. Дана матриця цілих чисел. Зібрати всі додатні елементи нижче побічної діагоналі (заповнення здійснювати паралельно побічній діагоналі).

Варіант 15. Дана матриця дійсних чисел. Знайти число з максимальною дробовою частиною, переставити рядки й стовпці так, щоб це число стояло в лівому верхньому куті .

Варіант 16. Дана дійсна матриця A(N,M). Скласти програму знаходження мінімального від'ємного елемента матриці й знаходження його місця розташування.

Варіант 17. Дана дійсна матриця A(N,M). Скласти програму знаходження максимального додатного елемента матриці й знаходження його місця розташування.

Варіант 18. Дана матриця дійсних чисел. Знайти мінімальний елемент і найбільш віддалений від нього (за значенням) елемент матриці. Пошук здійснювати в напрямку, паралельному побічній діагоналі.

Варіант 19. Дана матриця цілих чисел. Зібрати всі від'ємні елементи вище побічної діагоналі (заповнення здійснювати по стовпцях).

Варіант 20. Дана матриця дійсних чисел. Знайти мінімальний елемент і найближчий до нього (за значенням) елемент матриці. Пошук здійснювати у квадратному контурі , центром якого є мінімум, а довжина боку - п'ять елементів масиву.

Варіант 21. Дана матриця дійсних чисел. Знайти мінімальний елемент матриці. Пошук здійснювати в напрямку, паралельнім побічній діагоналі.

Варіант 22. Дана матриця дійсних чисел. Знайти максимальний елемент матриці. Пошук здійснювати в напрямку, паралельнім головній діагоналі.

Варіант 23. Дана матриця дійсних чисел. Знайти максимальний елемент і найближчий до нього (за значенням) елемент матриці. Пошук здійснювати в напрямку, паралельнім побічній діагоналі.

Варіант 24. Дана матриця дійсних чисел. Знайти число з мінімальною дробовою частиною, переставити рядки й стовпці так,щоб це число стояло в правому верхньому куті.

Варіант 25. Дана матриця дійсних чисел. Знайти максимальне й мінімальне додатні числа й кількість чисел між ними.

ЛАБОРАТОРНА РОБОТА №4
СОРТУВАННЯ МАСИВІВ

Ціль роботи: знайомство з методами сортування даних, найбільш часто застосовуваних на практиці.

1. Методичні вказівки до виконання роботи

Під сортуванням розуміють процес перестановки об'єктів даного множини в певному порядку. Ця мета може бути досягнена за допомогою різних алгоритмів, причому кожний з них має як свої переваги, так і свої недоліки .

Метод простого вибору

У початковій послідовності відшукується найменший елемент. Цей елемент записується у вихідний масив. Знайдене мінімальне значення позначається ознакою й не бере участь у подальшому сортуванні. Цей процес необхідно повторити N разів(N - розмірності початкового масиву).

Модифікований метод простого вибору

У послідовності чисел відшукується найменший елемент, який ставиться на перше місце. Для того, щоб не втратити елемент, що стоїть на першому місці, його встановлюють на місце мінімального, тобто знайдений мінімальний елемент і перший міняються місцями. Потім в усіченій послідовності (з розгляду виключається перший елемент) відшукується другий найменший елемент і ставиться на друге місце. Цей процес повторюється N-1 разів(N - розмір масиву, що сортується), поки не встане на своє місце передостанній N-1 елемент, зрушивши максимальний елемент у кінець послідовності.

Метод парних перестановок

Метод полягає в попарному порівнянні елементів і їх обміні, якщо вони стоять не в порядку зростання (убування). Перегляд усього масиву й порівняння пар повторюється доти, поки не будуть відсортовані всі елементи, тобто під час чергового перегляду не відбудеться ні однієї перестановки.

Метод "спливаючого пухирця"

Метод дістав свою назву в результаті наступної аналогії. Якщо елементи у вертикально розташованому масиві уявити собі пухирцями в резервуарі з водою, що мають вагу, рівну значенню елемента, то кожний перегляд масиву й перестановки елементів будуть розглядатися як "спливання" пухирця на відповідний його вазі рівень . У послідовності порівнюються два сусідні елементи, наприклад, К і К+1, припустимо, відбулася перестановка цих елементів. У цьому випадку далі робиться перевірка для всіх елементів від Кого до першого, тобто рух "нагору" по послідовності з перестановкою відповідних елементів.

Щоб не переглядати масив щораз із самого початку, можна запам'ятати місце (індекс) останнього обміну. Усі пари сусідніх елементів з індексами, меншими, ніж запам’ятований, уже розташовані в потрібному порядку.

Метод шейкер-сортування

У послідовності під час першого перегляду шукається мінімальний елемент і ставиться в початок масиву. Далі, під час другого перегляду усіченого масиву (виключено з розгляду перший елемент) знаходиться максимальний елемент і ставиться в кінець масиву. Потім наступним переглядом визначається мінімальний елемент і ставиться на своє місце і так далі, постійно чергуючи пошук мінімуму й максимуму й зменшуючи зону перегляду із країв.

Сортування методом Шелла

Сортування методом Шеллаподібне сортування методом парних перестановок у тому розумінні, що йде порівняння й, у потрібному випадку , перестановка пар елементів. Але порівнюються не сусідні елементи, а ті, які знаходяться один від одного на відстані D.

Ця величина спочатку рівна D=(N+l)/2 (N - кількість елементів послідовності, що сортується). Після кожного перегляду відстань між елементами (D) зменшується.

D(i+1)=floor((D(i)+1)/2)

Функція floor повертає округлене убік зменшення значення виразу. Останній перегляд здійснюється при D=1.

Сортування підрахунком

Суть методу полягає у визначенні місця (індексу) знаходження кожного елемента вихідного масиву у відсортованому масиві. Для цього кожний елемент вихідної послідовності порівнюється з усіма іншими елементами й підраховується, скільки елементів менше порівнюваного. Це значення визначає число елементів, які будуть знаходитися перед розглянутим елементом у відсортованому масиві , і визначає індекс елемента (індекси елементів масиву в мові CI починаються з 0). Індекси всіх елементів зберігаються в допоміжному масиві. Наприкінці роботи необхідно розставити елементи на місця відповідно до отриманих індексів.

2. Варіанти завдань

Варіант 1. Дана послідовність, розташувати її додатні елементи, що стоять на непарних місцях по зростанню.

Варіант 2. Дана послідовність, розташувати її ненульові елементи по убуванню.

Варіант 3. Переставити рядки вихідної матриці так, щоб убувала кількість нулів у рядках.

Варіант 4. Упорядкувати всі рядки матриці по числу елементів, кратних 3, тобто на перше місце поставити рядок з найменшим числом таких елементів і т.д. , на останнє місце - з найбільшим числом таких елементів.

Варіант 5. Розташувати в порядку зростання додатні елементи правого верхнього трикутника матриці.

Варіант 6. Розташувати в порядку убування додатні елементи лівого нижнього трикутника матриці.

Варіант 7. Дана послідовність, елементи якої є цілі двозначні числа. Упорядкувати послідовність за зростанням сум цифр відповідних елементів.

Варіант 8. Дана послідовність, розташувати її від'ємні елементи по убуванню.

Варіант 9. Дана послідовність, елементи якої є цілі двозначні числа. Упорядкувати послідовність по убуванню добутків цифр відповідних елементів.

Варіант 10. Дана послідовність, розташувати її елементи, що потрапляють в інтервал від A до B, у порядку зростання.

Варіант 11. Дана послідовність, розташувати її парні (за значенням) елементи по убуванню.

Варіант 12. Дана послідовність, розташувати її елементи, кратні 3, по убуванню.

Варіант 13. Дана матриця. Упорядкувати її рядки, що містять нулі, у порядку зростання їх кількості.

Варіант 14. Дана матриця. Упорядкувати по убуванню додатні елементи її правої половини.

Варіант 15. Дана матриця. Упорядкувати по зростанню ненульові елементи її нижньої половини.

Варіант 16. Дана послідовність чисел, розташувати її від'ємні елементи, що стоять на непарних місцях, по зростанню.

Варіант 17. Дана послідовність чисел, розташувати її ненульові елементи по зростанню.

Варіант 18. Переставити рядки вихідної матриці так, щоб зростала кількість нулів у рядках.

Варіант 19. Упорядкувати всі рядки матриці по числу елементів, кратних 2, тобто на перше місце поставити рядок з найменшим числом таких елементів і т.д. , на останнє місце - з найбільшим числом таких елементів.

Варіант 20. Розташувати в порядку зростання ненульові елементи правого верхнього трикутника матриці.

Варіант 21. Розташувати в порядку убування ненульові елементи лівого нижнього трикутника матриці.

Варіант 22.Дана послідовність, елементи якої є цілі тризначні числа. Упорядкувати послідовність по зростанню сум цифр відповідних елементів.

Варіант 23. Дана послідовність чисел, розташувати ІА додатні елементи по убуванню.

Варіант 24. Дана послідовність чисел, елементи якої є цілі тризначні числа. Упорядкувати послідовність по убуванню добутків цифр відповідних елементів.

Варіант 25. Дана послідовність, розташувати її елементи, що не потрапляють в інтервал від А до В, у порядку зростання.

ЛАБОРАТОРНА РОБОТА №5
ОБРОБКА РЯДКІВ

Ціль роботи: знайомство із програмними засобами опису й обробки даних типу рядок мовою С.

1. Методичні вказівки до виконання роботи

Рядки - це послідовність символів, взята в лапки. Наприкінці кожного рядка компілятор додає нульовий символ, що представляється керуючою послідовністю '\0\'.

Рядок описується як масив символів. Число елементів масиву дорівнює числу елементів у рядку плюс символ кінця рядка (\0). Символьний рядок у програмі може розташовуватися на декількох рядках. Для переносу використовується зворотна дробова риса з наступним натисканням клавіші введення. Зворотна дробова риса ігнорується компілятором,та наступний рядок, вважається продовженням попереднього.

Для роботи з рядками дуже зручно використовувати вказівники.

Приклад. Записати введений рядок символів у зворотному порядку.

```
#include<stdio.h>
voidmain() {
int top,bot;
char string[ 10],temp;  /*опис рядка як масиву символів*/
scanf("%s", string);
/* при введенні рядків символ & не використовується, тому що ім'я масиву є вказівником на його початок */
for(top=0,botr= 10; top<bot: top++,bot--)
  {
        temp=string[top]; string[top]=string[bot];
        string[bot]=temp;
   }
printf(" % s\n", string);}
```

Для введення одиночного символу із вхідного потоку використовується функція *getchar()*. Для виводу одиночного символу використовується функція *putchar(ch)*, де ch - виведений символ. Аргументом функції виводу може бути одиночний символ (включаючи знаки, що представляються керуючими послідовностями), змінна або функція, значенням якої є одиночний символ.

Приклад. Програма вводить із вхідного потоку один символ, а потім виводить його на екран.

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

```
voidmain()
{
charch;
ch=getchar();
putchar(ch);
}
```

Приклад. Програма, що реалізує читання й друк символів до введення знака «*».

```
#include<stdio.h>
#define STOP *
voidmain()
{
charch;
while((ch=getchar())!=STOP)   putchar(ch);}
```

В умові, щостоїть після ключового слова *while*, реалізоване відразу три дії: введення символу за допомогою функції *getchar()*; занесення символу в змінну ch; перевірка на ознаку кінця введення. Для реалізації перевірки на кінець введення послідовності символів використовується ідентифікатор STOP, визначений директивою препроцесора.

Для роботи з рядками є набір функцій. Для введення із стандартного пристрою введення (клавіатури) найчастіше використовуються бібліотечні функції з модуля стандартного введення - виводу: *scanfmagets*.

Для введення рядка за допомогою функції *scanf* використовують формат "%s", причому зверніть увагу на те, що перед ідентифікатором рядка не використовується знак адреси "&", тому що одномірний масив уже представлений вказівником на його початок:

```
char *s;
scanf("%s", s);
```

Функція gets() зчитує символи доти, поки не досягне символу переходу на новий рядок. Функція приймає всі символи аж до символу переведення рядка, але не включає його. До кінця рядка додається завершальний нуль ('\0').

Для виводу рядків можна використовувати дві функції: printf і puts. У функції printf у якості формату передається "%s", зручність при використанні цієї функції полягає в тому, що крім рядка можна відразу виводить дані інших типів. Особливість функції puts полягає в тому, що після виводу рядка автоматично відбувається перехід на наступний рядок.

Операції з рядками

Опис функції роботи з рядками <vars> міститься у файлі <string.h> бібліотеки стандартних функцій. Розглянемо деякі з них:

1. З'єднання послідовностей символів.

char *strcat(char *s1,char *s2)

Функція повертає вказівник на отриманий рядок.

2. Пошук першого входження символу в рядок.

char *strchr(char *s,int c)

Функція переглядає рядок (звертання до рядка за допомогою вказівника s) і шукає символ з кодом c. Повертає вказівник на знайдений символ або порожнє значення.

3. Порівняння рядків.

int strcmp(char *s1,char *s2)

Порівнюються два зазначені рядки. Результат - змінна типу int:

s 1 <s2 від'ємне значення

s 1 ==s2 значення 0

sl>s2 додатне значення

4. Копіювання символів.

char *strcpy(char *s1,char *s2)

Копіюється послідовність символів, зазначена параметром s1 за адресою s2.

5. Визначення довжини рядка (без завершального нуля).

int strlen(char *s)

6. Пошук входження однієї послідовності символів в іншу

char *strstr(const char *s1, const char *s2) - повертає вказівник на положення першої появи послідовності символів з s2 у рядку s1 (крім завершальних пробілів); повертає NULL, якщо збігів не знайдено.

7. Переформування рядка в окремі знаки

char *strtok(char *s1,constchar *s2) - ця функція переформує рядок s1 в окремі знаки; рядок s2 містить символи, які використовуються в якості роздільників. Функція викликається послідовно. Для першого виклику s1 повинен вказувати на рядок, який необхідно розбити на знаки. Функція знаходить роздільник, який іде за символом, що не є роздільником, і заміняє його пробілом. Вона повертає вказівник на рядок, що містить перший знак. Якщо жодного знака не знайдено, вона повертає NULL. Щоб знайти наступний знак у рядку, необхідно викликати функцію знову, але першим аргументом поставити NULL.

Кожний послідовний виклик повертає вказівник на наступний знак або на NULL, якщо більше знаків не знайдене.

Приклад поділу речення на окремі слова:

```
char s[80]; // Вихідне речення char
slova[20][20]; // Масив слів у реченні
char *<unk>p</unk>;
int sl; //Лічильник слів у реченні
char *r="!;:,.?-"; // Можливі роздільники між словами
gets(s); // Введення рядка
p=strtok (s, r);
while (p)
{
sl++;
strcpy(slova[sl],p);
p=strtok (NULL,r);
}
```

Перевірка символів

Для перевірки символів використовуються функції, що повертають значення "істина" або "хибність". Прототипи цих функцій описані у файлі <ctype.h> бібліотеки стандартних функцій. Розглянемо деякі з них:

Функція	*"Істина", якщо*
isalpha(c)	*c – символ алфавіту*
isupper(c)	*c – символ верхнього регістру*
islower(c)	*c – символ нижнього регістру*
isdigit(c)	*c – цифра від 0 до 9*
isxdigit(c)	*адцяткова цифра*
isalnum(c)	*c – літера або цифра*
isspace(c)	*c – пробіл, табуляція,* переведення рядків

Приклад. Програма перевіряє введений рядок і потім виводить на екран ту його частину, яка починається із символу 'y'.

```
#include<stdio.h>
 #include<string. h>
voidmain()
{
charstring[10];
```

```
char *ptr;
printf("Уведіть <vars>Введіть,Запровадьтерядок\n");
gets(string);
ptr=strchr(string,'y');
printf("це рядок %s\n",ptr);
}
```

2. Контрольні питання

1. Як описуються рядки мовою Сі?
2. Функції для введення й виводу символів.
3. Функції для введення й виводу рядків.
4. Функції перевірки символів.
5. Функції, що реалізують операції з рядками.
6. Ініціалізація рядків у програмі.
7. Використання вказівників для роботи з рядками.

3. Завдання

Варіант 1. Задане речення, слова в ньому розділені пробілом, підрахувати, скільки літер "а" у кожному слові.

Варіант 2. Задане речення, слова в ньому розділені пробілом , підрахувати, скільки літер і цифр в останньому слові.

Варіант 3. Задане речення, слова в ньому розділені пробілом , поміняти місцями перше й останнє слова.

Варіант 4. Задане речення, слова в ньому розділені пробілом , поміняти місцями парні й непарні один по одному проходження слова.

Варіант 5. Задані N речень. Знайти в кожному перше слово й надрукувати їх у рядок через пробіл.

Варіант 6. Задані N речень. Знайти в кожному останнє слово й надрукувати їх у рядок через пробіл.

Варіант 7. Задані N речень. Підрахувати кількість слів у кожному реченні й вивести на друк.

Варіант 8. Задане речення, слова в ньому розділені пробілом . Підрахувати кількість слів, які починаються з тієї літери, якою закінчується попереднє слово.

Варіант 9. Задане речення, слова в ньому розділені пробілом . Підрахувати кількість слів, які починаються з тієї ж літери, що й наступне слово.

Варіант 10. Задане речення, слова в ньому розділені пробілом . Упорядкувати слова в порядку зростання їх довжини.

Варіант 11. Задане речення, слова в ньому розділені пробілом. Упорядкувати слова за алфавітом (тільки по першій літері).

Варіант 12. Задані N речень, слова в яких розділені пробілами. Вивести їх на друк в порядку зростання кількості слів у реченні.

Варіант 13. Задані N речень, слова в яких розділені пробілами. Вивести їх на друк в порядку зростання спільної довжини слів у реченні (без урахування кількості поділяючих пробілів).

Варіант 14. Задане речення, слова в ньому розділені пробілом . Скласти з нього два речення за правилом: в одне переписати всі парні за порядком слідування слова, а в інше - непарні.

Варіант 15. Задані N речень, слова в яких розділені пробілом . Скласти новий текст за наступним правилом: виключити з тексту всі слова на літеру а'.

Варіант 16. Задане речення. Підрахувати, скільки разів зустрічається в кожному слові заданий символ.

Варіант 17. Задане речення. Роздрукувати всі літери B, перед якими безпосередньо знаходиться літера C в одному слові.

Варіант 18. Задане речення. Надрукувати true, якщо в заданім заданому слові літера **a** зустрічається частіше, ніж літера **b**, і надрукувати false у протилежному випадку.

Варіант 19. Задані N речень, слова в яких розділені пробілами. Вивести їх на друк в порядку убування кількості слів у реченні. Варіант 20. Задане речення, слова в ньому розділені пробілом . Скласти з нього два речення за правилом: у перше переписати всі непарні за порядком слідування слова, а в друге – парні.

Варіант 21. У реченні знайти всі однокореневі слова. Корінь задається із клавіатури.

Варіант 22. Видалити в реченні всі повторні входження слів і роздрукувати речення, що вийшло.

Варіант 23. Виділити з введеного речення слова, що містять повторювані літери.

Варіант 24. Усі літери кожного слова в реченні записати у зворотному порядку й роздрукувати речення, що вийшло.

Варіант 25. Перевірте на збіг два речення. Кількістю пробілів між словами нехтувати. Розділові знаки враховувати.

ЛАБОРАТОРНА РОБОТА №6
ОБРОБКА СТРУКТУР ДАНИХ

Ціль роботи: знайомство з описом структур даних мовою С, одержання практичних навичок обробки структур з використанням покажчиків.

1. Методичні вказівки до виконання роботи

Записи (структури) є одними з основних структур даних у мовах програмування високого рівня. Поняття запису використовується при машинній обробці різних документів, таблиць, баз даних.

Запис - це структура, що полягає з фіксованого числа компонентів, називаних полями. В одному полі дані мають той самий тип, а в різних полях можуть мати різні типи, за винятком функцій.

```
struct{
        type1 id11,id12,...,id1n;
        type2 id21,id22,...,id2 m;
        ……..……..…
        type iidk1,idk2,...,idkp;
} описуватель [описуватель];
```

Тут idij - ідентифікатори полів; type і - типи полів; описуватель - ім'я змінної із заданою структурою.

Приклад. Опис змінних datel і date2. Кожна змінна містить два поля.

```
struct{
    intyear;
    short day;
    } date1, date2;
```

Для опису структури зручно використовувати шаблони. Опис шаблону йде без наступного списку змінних.

Формат шаблону наступний

struct ім'я_типу_структури{список описів;};

Опис шаблону є описом нового типу даних. Далі можна описувати змінні, використовуючи ім'я шаблону.

Приклад. Опис шаблону для дати (день, місяць, рік).

```
struct data{
        int day;
        char month[10];
        int year;
      };
structdatad1 ,d2,d3; /*опис змінних */
```

При описі можлива ініціалізація змінної.

struct data d1={4,"Сентябрь",1998}; Можна задавати ім'я типу структури за допомогою описувача типу typedef.

Описувач типу

Цей описувач дозволяє створити своє власне ім'я типу.

Формат описувача типу:

typedefспецифікатор_типу описувача;

специфікатор типу — це основний або похідний тип даних або тип, який раніше визначений програмістом. Описувач - це нове ім'я створеного нами типу.

Приклад. Опис нового типу даних "дата", що полягає їх трьох полів.

```
typedef struct{
            int day;
            char month[10];
            int year;
            } data;
      data d1,d2,d3;
```

Структура не може містити в якості елемента структуру такого ж типу, але може включати покажчик на структуру цього типу, за умови, що в оголошенні структури зазначене ім'я типу. Це дозволяє створювати зв'язані списки структур.

Приклад. Опис бінарного дерева.

```
struct tree{
            int number;
            struct tree *left;
            struct tree *right;
           };
```

Доступ до елемента структури здійснюється за допомогою символу "." одержання, що позначає операцію, елемента структури.

Приклади звертання до елементів структур, описаних вище:

d1.day

d2.year

Доступ до елементів структур може здійснюватися й за допомогою покажчиків.

Приклад. Опис покажчика на структуру й звертання до її елементів.

```
data* ptr;
ptr=&d2;
```

```
ptr->day=4;
ptr->month=”Грудень”;
ptr->year=1998;
```

Звертання до елементів структури можна записати ще в такий спосіб:

```
(*ptr).day=4;
(* ptr).month="Декабрь";
(*ptr).year=1998;
```

Приклад. Опис масиву, елементами якого є структури типу

data. data d[5];

Приклад. Програма, що реалізує введення списку студентів і висновок інформації про студента по його номеру в списку. Уведення інформації про студента реалізований у вигляді окремої функції.

```
#include<stdio.h>
constin t n=3;
typedef struct{
                int num;
                char fam[20];
                char name[15];
                } student;
student mas[n];
student *ptr=&mas[0];
void vvod(student *p)
        {
        scanf("%d",&p->num);
        scanf("%s",p->fam);
        scanf("%s",p->name);
        }
void main()
{
int i,d;
for(i=0;i<n;i++)
vvod(ptr+i);
printf(“Уведіть номер студента в списку \n”);
scanf("%d",&d);
for(i=0;i<n;i++)
if(mas[i].num==d) printf(“Студент %s %s\n”,mas[i].fam,mas[i].name);
}
```

Процедурі vvod передається значення покажчика на елемент масиву mas, який є структурою. У наведеній програмі використано обоє способу звертання до елемента структури (з використанням покажчика й без нього).

2. Контрольні питання

1. Способи опису структури.
2. Описувач типу.
3. Способи звертання до елемента структури.
4. Опис масиву структур.

3. Завдання

Скласти модель наступного об'єкта:

Варіант 1. Склад студентів факультету з розбивкою на групи. Кількість спеціальностей на факультеті, груп кожної спеціальності й студентів у кожній групі задати самостійно. Скласти модуль пошуку адреси (посилання) елемента списку по його інформаційних полях.

Варіант 2. Склад спеціальностей вузу з розбивкою на факультети. Кількість факультетів і спеціальностей кожного факультету задати самостійно. Скласти модуль пошуку елементів з діапазоном шифрів спеціальностей від ШИФР1 до ШИФР2.

Варіант 3. Список внутрішніх телефонів організації з розбивкою по відділах. Кількість відділів і телефонів усередині відділу задати самостійно. Скласти модуль пошуку всіх телефонів із заданими двома першими цифрами.

Варіант 4. Список ділянок підприємства з розбивкою по цехах. Кількість цехів і ділянок кожного цеху задати самостійно. Скласти модуль пошуку ділянки з максимальним номером.

Варіант 5. Список працівників цеху з розбивкою по професіях. Кількість професій і працівників кожної професії цеху задати самостійно. Скласти модуль сортування прізвищ за алфавітом.

Варіант 6. Склад парку ЕОМ обчислювального центру з розбивкою ЕОМ по серіях. Кількість серій і ЕОМ різних серій задати самостійно. Скласти модуль сортування ЕОМ по зростанню обсягу оперативної пам'яті.

Варіант 7. Список номерів і маршрутів рейсів автобусів з розбивкою рейсів по районах проходження. Кількість районів і рейсів у кожний район задати самостійно. Скласти модуль сортування рейсів по убуванню номерів.

Варіант 8. Список ПЗ на лазерних дисках в обчислювальному центрі з розбивкою по операційних системах. Кількість ОС і дисків для кожної

системи задати самостійно. Скласти модуль пошуку елементів списку по його інформаційних полях.

Варіант 9. Склад спортивного змагання з розбивкою по видах спорту. Кількість видів спорту й учасників кожного виду спорту задати самостійно. Скласти модуль формування нового списку спортсменів, що містить прізвища кожного другого спортсмена.

Варіант 10. Програма роботи конференції з розбивкою доповідей по секціях. Кількість секцій і доповідей у кожній секції задати самостійно. Скласти модуль формування нового списку, що містить доповіді з декількома авторами.

Варіант 11. Конструкція технічного обладнання, що полягає з декількох блоків. Причому кожний блок містить певна кількість модулів. Скласти модуль формування нового списку, з якого віддаляється модуль із заданим найменуванням.

Варіант 12. Конструкція модуля, що полягає з мікросхем. Кількість мікросхем і виводів кожної мікросхеми задати самостійно. Скласти модуль формування нового списку введення-виведення, з якого віддаляються елементи з номерами 7 і 14.

Варіант 13. Конструкція обчислювальної мережі, що полягає з вузлів різних типів. Кількість вузлів і типів вузлів задати самостійно. Скласти модуль формування нового списку вузлів, у якім доданий новий заданий вузол.

Варіант 14. Книга складається із глав і параграфів. Кількість глав і параграфів кожної глави задати самостійно. Скласти модуль формування нового списку параграфів, у якім перший і останній елементи помінялися місцями.

Варіант 15. Задачники складаються зі списку завдань, розбитих на теми. Кількість тем і завдань задати самостійно. Скласти модуль формування нового списку завдань, у якім після завдання із заданим номером записаний елемент із новим завданням.

Варіант 16. Набір матеріалів, наявних на складі, з розбивкою по виду товару. Кількість видів товарів і штук кожного товару задати самостійно. Скласти модуль формування двох нових списків з вихідного списку товарів. У перший список попадають номера товарів менше деякого заданого, у другий - більше.

Варіант 17. Список школярів, що займаються в кружках, з розбивкою по кухлях. Кількість кружків і школярів у кожному кружку задати самостійно. Скласти модуль формування двох нових списків зі списку

школярів по наступному принципу: у перший список попадають перші десять школярів, у другий - інші.

Варіант 18. Список покупок, зроблених кимсь протягом місяця, з розбивкою по статтях витрат. Кількість статей витрати й покупок по кожній статті задати самостійно. Скласти модуль формування нового списку покупок із ціною більше заданої ЦІНИ.

Варіант 19. Список кварталів міста з розбивкою по районах. Кількість районів і кварталів у кожному районі задати самостійно. Скласти модуль формування нового списку кварталів, зведеного для двох зазначених районів.

Варіант 20. Список навчальних дисциплін, які повинен вивчити студент за час навчання у вузі з розбивкою на цикли (апаратний, загальнонауковий, гуманітарний і т.д.). Кількість циклів і дисциплін у циклах для різних спеціальностей задати самостійно. Скласти модуль формування нового списку, отриманого злиттям трьох списків дисциплін для заданих циклів.

Варіант 21. Список деталей механічного обладнання з розбивкою деталей по агрегатах. Кількість агрегатів і деталей у кожному агрегаті задати самостійно. Скласти модуль формування нового списку деталей, загальних для двох заданих агрегатів.

Варіант 22. Список номерів автомобілів, які парку валися на платній стоянці, з розбивкою по марках. Кількість марок і автомобілів кожної марки задати самостійно. Скласти модуль формування нового списку, що містить номера із заданою частиною номера.

Варіант 23. Список підприємств, розташованих у місті, з розбивкою по міністерствах. Кількість міністерств і підприємств по кожнім міністерству задати самостійно. Скласти модуль формування нового списку підприємств, що є заводами.

Варіант 24. Список вузів країни з розбивкою по профілях. Кількість профілів і вузів кожного профілю задати самостійно. Скласти модуль формування нового списку вузів заданого профілю, що є університетами.

Варіант 25. Список співробітників підприємства з розбивкою по посадах. Кількість посад і кількість співробітників, що мають однакову посаду, задати самостійно. Скласти модуль формування списку однофамільців, що працюють на одній посаді.

ЛАБОРАТОРНА РОБОТА №7
ОБРОБОТКА СПИСКІВ

Ціль роботи: придбати практичні навички роботи з динамічними структурами даних мовою Сі.

1. Методичні вказівки до виконання роботи

Список - ця безліч елементів, кожний з яких полягає, як мінімум, із двох полів. Одне поле містить або саму інформацію, або посилання на неї. Інше поле містить посилання на наступний елемент списку. Елемент списку називають "ланка" списку. Таким чином, список - це ланцюжок зв'язаних ланок від першого до останнього. Остання ланка не посилається на наступний елемент, тому поле посилання має значення "порожній покажчик". По односпрямованому спискові можна рухатися тільки в одному напрямку - від заголовного (першого) ланки до останнього.

Двонаправлений (двозв’язний) список - це безліч елементів, кожний з яких має два поля з покажчиками; одне поле містить посилання на наступний елемент, інше поле - посилання на попередній елемент і інформаційне поле.

Наявність посилань як на наступну ланку, так і на попереднє дозволяє від кожної ланки рухатися за списком у будь-якому напрямку.

Список, перша ланка якого має посилання на останнє, а останнє на перше, називається кільцевий. Кільцевий список має заголовна ланка. Заголовна ланка, як і у випадку односпрямованого списку, дозволяє обробляти перше й останнє ланки в загальному циклі. Однак у такому кільцевому списку треба щораз перевіряти, не чи є чергова ланка заголовним.

Характерною рисою динамічних даних є неможливість установити заздалегідь фіксований обсяг пам'яті. Виділення пам'яті під окрему ланку списку відбувається в той момент, коли вона з'являється під час виконання програми, а не під час трансляції.

Керування оперативною пам'яттю реалізується за допомогою бібліотечних функцій мови Сі,що втримуються у файлі<alloc.h>: malloc, calloc, free. Функції malloc і calloc здійснюють виділення пам'яті, а функція free— її звільнення.

Функція malloc (unsignedsize) повертає в якості свого значення покажчик на область пам'яті. Розмір цієї області в байтах визначається за допомогою аргументу size. Якщо виділення обсягу пам'яті неможливо, то результатом функції буде порожній покажчик.

Функція calloc(unsignedcount, size) виділяє обнулену область пам'яті, у якій можна розмістити count об'єктів розміром size кожний. Результат функції - покажчик на виділену область пам'яті.

Функція *free(ptr)* звільняє область пам'яті, виділену раніше, обумовлену за допомогою аргументу ptr.

В C++ визначена також функція new для динамічного виділення пам'яті. Приклад її використання:

```
int *ptr;
ptr=new int; //виділня пам’яті розміром
int delete ptr; // очищення виділенної пам’яті
```

Приклад. Програма, що створює лінійний односпрямований список із прізвищ студентів.

```
#include<alloc.h>
 #include<stdio. h>
void main()
{typede fstruct man{ char name[20];
                    man *next; } man;
man *first, *cur;
int  n;
printf("введіть кількість імен ");
scanf("%d",&n);
first=(man *)malloc(sizeof(man));
cur=first;
for(int i=0;i<n;i++)
{if (i) {(*cur).next=(man *)malloc(sizeof(man));
            cur=(*cur).next;}
printf(" введітьим’я ");
      scanf("%s",(*cur).name);
      (*cur).next=NULL;
}
/* перегляд і висновок */
cur=first;
while (cur!=NULL)
{
      printf("Это %s\n",(*cur).name);
      cur=(*cur).next;
}
}
```

Дана програма спочатку запитує кількість імен у списку, а потім формує його, заповнюючи інформаційне поле.

Найбільш раціональної є програма, де формування списку відбувається залежно від відповіді на запит про припинення введення інформаційних полів.

Приклад двозв'язного списку - список делегатів від міст. Кількість міст і делегатів від кожного міста буде довільним.

```
typedef struct deputat
{char *name;
deputat *next;
};
typedef struct town
{char *nametown;
town *nexttown;
deputat *spisok;
};
```

2. Контрольні питання

1. Що таке динамічні дані?
2. Що таке список, види списків?
3. Створення кільцевого списку.
4. Функції керування оперативною пам'яттю.

3. Завдання

Частина 1. **Скласти динамічну модель наступного об'єкта:**

Варіант 1. Склад студентів факультету з розбивкою на групи. Кількість спеціальностей на факультеті, груп кожної спеціальності й студентів у кожній групі змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль пошуку адреси (посилання) елемента списку по його інформаційних полях.

Варіант 2. Склад спеціальностей вузу з розбивкою на факультети. Кількість факультетів і спеціальностей кожного факультету змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль пошуку елементів з діапазоном шифрів спеціальностей від ШИФР1 до ШИФР2.

Варіант 3. Список внутрішніх телефонів організації з розбивкою по відділах. Кількість відділів і телефонів усередині відділу змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль пошуку всіх телефонів із заданими двома першими цифрами.

Варіант 4. Список ділянок підприємства з розбивкою по цехах. Кількість цехів і ділянок кожного цеху змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль пошуку ділянки з максимальним номером..

Варіант 5. Список працівників цеху з розбивкою по професіях. Кількість професій і працівників кожної професії цеху змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль сортування прізвищ за алфавітом.

Варіант 6. Склад парку ЕОМ обчислювального центру з розбивкою ЕОМ по серіях. Кількість серій і ЕОМ різних серій змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль сортування ЕОМ по зростанню обсягу оперативної пам'яті.

Варіант 7. Список номерів і маршрутів рейсів автобусів з розбивкою рейсів по районах проходження. Кількість районів і рейсів у кожний район змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль сортування рейсів по убуванню номерів.

Варіант 8. Список ПО на лазерних дисках в обчислювальному центрі з розбивкою по операційних системах. Кількість ОС і дисків для кожної системи змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль пошуку елементів списку по його інформаційних полях.

Варіант 9. Склад спортивного змагання з розбивкою по видах спорту. Кількість видів спорту й учасників кожного виду спорту змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку спортсменів, що містить прізвища кожного другого спортсмена.

Варіант 10. Програма роботи конференції з розбивкою доповідей по секціях. Кількість секцій і доповідей у кожній секції змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку, що містить доповіді з декількома авторами.

Варіант 11. Конструкція технічного обладнання, що полягає з декількох блоків. Причому кожний блок містить довільна кількість модулів. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку, з якого віддаляється модуль із заданим найменуванням.

Варіант 12. Конструкція модуля, що полягає з мікросхем. Кількість мікросхем і виводів кожної мікросхеми змінно. Для фрагмента моделі, що є

одномірним списком, скласти модуль формування нового списку виводів, з якого віддаляються елементи з номерами 7 і 14.

Варіант 13. Конструкція обчислювальної мережі, що полягає з вузлів різних типів. Кількість вузлів і типів вузлів змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку вузлів, у якім доданий новий заданий вузол.

Варіант 14. Книга складається із глав і параграфів. Кількість глав і параграфів кожної глави змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку параграфів, у якім перший і останній елементи помінялися місцями.

Варіант 15. Задачники складаються зі списку завдань, розбитих на теми. Кількість тем і завдань змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку завдань, у якім після завдання із заданим номером записаний елемент із новим завданням.

Варіант 16. Набір матеріалів, наявних на складі, з розбивкою по виду товару. Кількість видів товарів і штук кожного товару змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування двох нових списків з вихідного списку товарів. У перший список попадають номера товарів менше деякого заданого, у другий - більше.

Варіант 17. Список школярів, що займаються в кружках, з розбивкою по кухлях. Кількість кружків і школярів у кожному кружку змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування двох нових списків зі списку школярів по наступному принципу: у перший список попадають перші десять школярів, у другий - інші.

Варіант 18. Список покупок, зроблених кимсь протягом місяця, з розбивкою по статтях витрат. Кількість статей витрати й покупок по кожній статті змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку покупок із ціною більше заданої ЦІНИ.

Варіант 19. Список кварталів міста з розбивкою по районах. Кількість районів і кварталів у кожному районі змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку кварталів, зведеного для двох зазначених районів.

Варіант 20. Список навчальних дисциплін, які повинен вивчити студент за час навчання у вузі, з розбивкою на цикли (апаратний, загальнонауковий, гуманітарний і т.д.). Кількість циклів і дисциплін у циклах для різних спеціальностей змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку, отриманого злиттям трьох списків дисциплін для заданих циклів.

Варіант 21. Список деталей механічного обладнання з розбивкою деталей по агрегатах. Кількість агрегатів і деталей у кожному агрегаті змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку деталей, загальних для двох заданих агрегатів.

Варіант 22. Список номерів автомобілів, які пакуються на платній стоянці, з розбивкою по марках. Кількість марок і автомобілів кожної марки змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку, що містить номера із заданою частиною номера.

Варіант 23. Список підприємств, розташованих у місті, з розбивкою по міністерствах. Кількість міністерств і підприємств по кожнім міністерству змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку підприємств, що є заводами.

Варіант 24. Список вузів країни з розбивкою по профілях. Кількість профілів і вузів кожного профілю змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування нового списку вузів заданого профілю, що є університетами.

Варіант 25. Список співробітників підприємства з розбивкою по посадах. Кількість посад і кількість співробітників, що мають однакову посаду, змінно. Для фрагмента моделі, що є одномірним списком, скласти модуль формування списку однофамільців, що працюють на одній посаді.

Частина 2. **Додаткові завдання**.

Варіант 1. Дана послідовність символів, що кінчаються крапкою. Знайти всіх "сусідів" заданого символу. Перший і останній символ уважати "сусідами".

Варіант 2. Дана послідовність символів, що кінчаються крапкою. Підрахувати кількість символів, у яких однакові "сусіди". Перший і останній символ уважати "сусідами".

Варіант 3. Дана послідовність символів, що кінчаються крапкою. Вилучити всі символи, у яких однакові "сусіди". Перший і останній символ уважати "сусідами".

Варіант 4. Дана послідовність символів, що кінчаються крапкою. Переставити у зворотному порядку всі символи між першим і останнім входженнями заданого символу (якщо символ входить у послідовність не менш двох раз).

Варіант 5. Дана послідовність символів. У кінець даної послідовності додати всі її символи, розташовуючи їх у зворотному порядку. Наприклад, з послідовності символів 1 2 3 одержати кільцевий список 1 2 3 2 1.

Варіант 6. Дана послідовність латинських букв, що кінчаються крапкою. Нехай символ k означає скасування попередньої букви; n символів підряд скасовують (стирають) n попередніх букв, якщо вони є. Перетворити послідовність із урахуванням входження в неї символу k.

Варіант 7. Даний запис багаточлена від зміною x довільному ступеня с цілими коефіцієнтами, причому його одночлени можуть бути й не впорядковані по ступенях x , а одночлени однієї й тієї ж ступеня можуть повторюватися. Наприклад,, $8*x^4-15*x+5*x^4-x^2+5-x$.Привести подібні члени в цьому багаточлені.

Варіант 8. Даний запис багаточлена від зміною x довільному ступеня із цілими коефіцієнтами, причому його одночлени можуть бути й не впорядковані по ступенях x , а одночлени однієї й тієї ж ступеня можуть повторюватися. Наприклад,$8*x^4-15*x+5*x^4-x^2+5-x$.Розташувати одночлени по убуванню ступенів x.

Варіант 9. Дане натуральне число n і дійсні числа $x_1,x_2,\ldots x_n$. За допомогою кільцевого списку обчислити $x_1*x_n+x_{2*}x_{n-1}+..+x_n* x_1$.

Варіант 10. Дане натуральне число n і дійсні числа $x_1,x_2,\ldots x_n$. За допомогою кільцевого списку обчислити$(x_1+x_n)*(x_2+ x_{n-1})*..*(x_n+ x_{1)}$.

Варіант 11. Дане натуральне число n і дійсні числа $x_1,x_2,\ldots x_n$. За допомогою кільцевого списку обчислити $(x_1+x_2+2*x_n)*(x_2+x_3+ 2*x_{n-1})*..*(x_{n-1}+x_n+2* x_{1)}$.

Варіант 12. Дане натуральне число n і дійсні числа$x_1,x_2,\ldots x_n$. Одержати кільцевий список виду$x_1,x_2,\ldots x_n, x_1,x_2,\ldots x_n$.

Варіант 13Дане натуральне число n і дійсні числа$x_1,x_2,\ldots x_n$. Одержати кільцевий список виду$x_1,x_2,\ldots x_n, x_n,x_{n-1},\ldots x_1$.

Варіант 14.Дане натуральне число n і дійсні числа$x_1,x_2,\ldots x_n$. Одержати кільцевий список виду $x_n,x_{n-1},\ldots x_1,x_1,x_2,\ldots x_n, x_n$.

Варіант 15. Дана послідовність латинських букв, що кінчаються крапкою. Нехай символ k означає скасування наступної букви; n символів підряд скасовують (стирають) n наступних букв, якщо вони є. Перетворити послідовність із урахуванням входження в неї символу k.

Додаток А. Таблиця деяких математичних функцій

Позначення	Тип аргументу	Назва
abs(x)	int	модуль
fabs(x)	double	модуль
labs(x)	long	модуль
cos(x)	double	косинус
sin(x)	double	синус
tan(x)	double	тангенс
sqrt(x)	double	корінь квадратний
pow(x,y)	x,y- double	X у ступені У
ceil(x)	double	округлення убік збільшення
floor(x)	double	округлення убік зменшення
fmod(x,y)	x,y- double	визначення залишку від розподілу x на y
exp(x)	double	експонента
log(x)	double	натуральний логарифм
log10(x)	double	десятинний логарифм
modf(x,y)	double	поділ числа на цілу й дробову частини

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Керниган, Б.,Мова програмування Сі. Завдання по мові Сі/Б.Керниган, Д.Ритчи. —М.:ФиС, 1985. — 280 с.

2. Страустрап, Б. Мова програмуванняСі++/Б.Страуструп. — М.:Радіо та зв'язок, 1991.—352 с.

3. Белецкий, Я. Енциклопедія мови Сі /Я.Белецкий. — М.:Мир, 1992. — 687 с.

4. Белецкий, Я.ТурбоСі++:Нова розробка: навч. посібник для студентів вищих навчальних закладів / Я.Белецкий. — М.'Машинобудівництво, 1994. — 400с.

5. Пильщиков, В. Н. Збірник вправ по мові Паскаль: навч. посібник для втузов /В. Н.Пильщиков. — М.:Висш. шк.,1990. —223 с.

Зміст

Навчальне видання
Методичні вказівки до виконання лабораторних робіт з дисципліни «Інформатика» (Частина 2)
Для студентів напряму 6.050903 «Телекомунікації»
(денної і заочної форм навчання)

Укладачі:

Яремко Ігор Миколайович, старший викладач
Долгіх Ірина Петрівна, старший викладач

Затверджені на засіданні Навчально-видавничої ради ДонНТУ
Протокол № 4 від 07.10.2010 Р.№344 3,13 друк.арк.