МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

**С. Б. ПРИХОДЬКО, Л. М. МАКАРОВА,
Т. Г. СМИКОДУБ**

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни "Основи програмування"

У двох частинах

Частина 2

Рекомендовано Методичною радою НУК

Електронне видання
комбінованого використання на DVD-ROM

МИКОЛАЇВ • НУК • 2018

УДК 004.4(076)
П 77

Автори: С. Б. Приходько, д-р техн. наук, професор;
Л. М. Макарова, канд. техн. наук;
Т. Г. Смикодуб, старш. викладач

Рецензент І. І. Коваленко, д-р техн. наук, професор

Приходько С. Б.

П 77 Методичні вказівки до виконання лабораторних робіт з дисципліни "Основи програмування" : у 2 ч. Ч. 2 / С. Б. Приходько, Л. М. Макарова, Т. Г. Смикодуб. – Миколаїв : НУК, 2018. – 127 с.

Уміщено завдання для лабораторних робіт, стислий виклад теоретичних відомостей, етапи виконання робіт і контрольні питання.

Призначено для студентів першого курсу спеціальності 121 "Інженерія програмного забезпечення" (попередня назва "Програмне забезпечення систем"), а також для усіх, хто вивчає основи програмування мовою С++.

УДК 004.4(076)

Навчальне видання

ПРИХОДЬКО Сергій Борисович **МАКАРОВА** Лідія Миколаївна
СМИКОДУБ Тетяна Георгіївна

**МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
"Основи програмування"**

У двох частинах

Частина 2

Комп'ютерне верстання та складання *В. В. Москаленко*
Коректор *М. О. Паненко*

Формат 60×84/16. Ум. друк. арк. 8,1. Об'єм даних 2068 кб.
Тираж 15 прим. Вид. № 15. Зам. № 153.

Видавець і виготівник Національний університет кораблебудування
імені адмірала Макарова
просп. Героїв України, 9, м. Миколаїв, 54025
E-mail : publishing@nuos.edu.ua

Свідоцтво суб'єкта видавничої справи ДК № 2506 від 25.05.2006 р.

ВСТУП

У Національному університеті кораблебудування імені адмірала Макарова (НУК) дисципліна "Основи програмування" вивчається студентами спеціальності 121 "Інженерія програмного забезпечення" (попередня назва "Програмне забезпечення систем") у першому-третьому семестрах. Вона відноситься до циклу професійної та практичної підготовки бакалаврів.

Далі методичні вказівки мають допомогти студентам першого курсу спеціальності 121 "Інженерія програмного забезпечення" у підготовці та виконанні лабораторних робіт з дисципліни "Основи програмування". Вони містять завдання для лабораторних робіт, стислий виклад теоретичних відомостей, етапи виконання робіт та контрольні запитання.

При виконанні кожної лабораторної роботи рекомендується:

- засвоїти теоретичні відомості;
- розібратися з завданням та етапами виконання роботи;
- виконати запропоновані завдання та оформити звіт з роботи;
- перевірити повноту розуміння теми за допомогою контрольних запитань, розташованих у кінці кожної роботи.

Лабораторну роботу необхідно виконувати відповідно до наведених етапів виконання роботи. Звіт про лабораторну роботу оформлюється кожним студентом індивідуально. Звіт повинен містити: назву і мету роботи; завдання (постановку задачі); методику і алгоритм розв'язання задачі; текст програми; результати роботи; аналіз результатів та висновки.

Робота № 1

Розробка і реалізація програм пошуку та сортування

Мета роботи: оволодіння навичками складання програм пошуку і сортування елементів масиву та виконання їх в NetBeans IDE.

Завдання

Завдання 1.1. Сформулювати математичний запис фрагмента програми та обчислити значення змінної `x` після його виконання, згідно варіанту, наведеному в таблиці 1.1. Елементи масиву обчислюються за формулою `a[i]=p[i]-50`, де `p[i+1]=(p[i]*11+7)%100.` `p[0]` дорівнює `N` - номеру варіанта за списком групи, кількість елементів у масиві дорівнює `size=15.`

Таблиця 1.1 – Варіанти завдання 1.1

№	Фрагмент програми

1–5	`for (int i=0; i< size; i++) {`
	`for (int j= size-1; j>i; j--){`
	`if (a[j] < a[j-1]) {`
	`int t = a[j];`
	`a[j] = a[j-1];`
	`a[j-1] = t;`
	`}`
	`}`
	`}`
	`x=a[0];`

6–10	`for (int i=1; i<size; i++) {`
	`int curr = a[i];`
	`int j = i-1;`
	`while (j>=0 && a[j]>curr) {`
	`a[j+1]=a[j];`
	`j--;`

Продовж. табл. 1.1

№	Фрагмент програми

	}
	a[j+1] = curr;
	}
	x=a[0];

11–15	for (int i=0; i<size-1; i++) {
	int nmin = i;
	for (int j=i+1; j<size; j++){
	if (a[j]<a[nmin]) nmin = j;
	}
	if (i != nmin) {
	int t = a[nmin];
	a[nmin] = a[i];
	a[i] = t;
	}
	}
	x=a[0];

16–20	for (int i=0; i< size; i+=2){
	for (int j= size-1; j>i; j-=2){
	if (a[j] > a[j-2]) {
	int t = a[j];
	a[j] = a[j-2];
	a[j-2] = t;
	}
	}
	}
	x=a[2];

21–25	for (int i=0; i<size-1; i++) {
	int nmin = i;
	for (int j=i+1; j<size; j++){
	if (a[j]>a[nmin]) nmin = j;
	}
	if (i != nmin) {
	int t = a[nmin];
	a[nmin] = a[i];
	a[i] = t;
	}
	}
	x=a[0];

№	Фрагмент програми

26–30	`for (int i=1; i<size; i+=2) {`
	`  int curr = a[i];`
	`  int j = i-2;`
	`  while (j>=0 && a[j]>curr) {`
	`  a[j+2]=a[j];`
	`  j-=2;`
	`  }`
	`  a[j+2] = curr;`
	`  }`
	`x=a[1];`

Завдання 1.2. Скласти програму сортування послідовностей чисел згідно за варіантами, які наведені в таблиці 1.2, та виконати її в NetBeans IDE. Послідовності заповнити за допомогою датчика випадкових чисел.

Таблиця 1.2 – Варіанти завдання 1.2

№	Завдання
1	Дана послідовність цілих чисел $a_1, a_2, \ldots, a_{25}$. Розташувати елементи послідовності, кратні 3 за зростанням (інші елементи залишаються на своїх місцях). Метод сортування – вибір.
2	Дана послідовність дійсних чисел $a_1, a_2, \ldots, a_{22}$. Розташувати додатні елементи послідовності, що стоять на непарних місцях за зростанням (інші елементи залишаються на своїх місцях). Метод сортування – вибір.
3	Дана послідовність дійсних чисел $a_1, a_2, \ldots, a_{22}$. Елементи, що стоять на парних місцях, розташувати в порядку зростання, а на непарних в порядку зменшення. Метод сортування – бульбашкою.
4	Дана послідовність дійсних чисел $a_1, a_2, \ldots, a_{22}$. Розташувати елементи послідовності, що стоять на парних місцях у порядку зменшення (інші елементи залишаються на своїх місцях). Метод сортування – бульбашкою.

№	Завдання
5	Дана послідовність дійсних чисел $a_1, a_2, \dots, a_{22}$. Розташувати елементи послідовності, менші за середньоарифметичне значення, за зростанням (інші елементи залишаються на своїх місцях). Метод сортування – вставка.
6	Дана послідовність дійсних чисел $a_1, a_2, \dots, a_{22}$, відмінних від нуля. Розташувати від'ємні елементи послідовності за спаданням, а додатні – по зростанню. Метод сортування – вибір.
7	Дана послідовність додатних цілих чисел $a_1, a_2, \dots, a_{25}$. Розташувати елементи послідовності, що кратні 3, за спаданням (інші елементи залишаються на своїх місцях). Метод сортування – вставка.
8	Дана послідовність цілих чисел $a_1, a_2, \dots, a_{25}$. Розташувати парні елементи за зростанням (інші елементи залишаються на своїх місцях). Метод сортування – вибір.
9	Дана послідовність дійсних чисел $a_1, a_2, \dots, a_{22}$. Потрібно розташувати додатні елементи послідовності за спаданням (інші елементи залишаються на своїх місцях). Метод сортування – вставка.
10	Дана послідовність цілих чисел $a_1, a_2, \dots, a_{25}$. Розташувати непарні елементи за спаданням (інші елементи залишаються на своїх місцях). Метод сортування – вибір.
11	Дана послідовність дійсних чисел $a_1, a_2, \dots, a_{22}$. Розташувати елементи послідовності, які належать діапазону $[p_1, p_2]$, за спаданням (інші елементи залишаються на своїх місцях). Метод сортування – бульбашкою.
12	Дана послідовність дійсних чисел $a_1, a_2, \dots, a_{30}$. Розташувати першу половину елементів послідовності, за спаданням, інші елементи розташувати за зростанням. Метод сортування – бульбашкою.
13	Дана послідовність дійсних чисел $a_1, a_2, \dots, a_{22}$. Розташувати елементи послідовності, які містять цифру N, за зростанням (інші елементи залишаються на своїх місцях). Метод сортування – вставка.

Продовж. табл. 1.2

№	Завдання
14	Дана послідовність дійсних чисел $a_1, a_2, \ldots, a_{20}$. Розташувати елементи послідовності, сума цифр яких дорівнює N, за спаданням (інші елементи залишаються на своїх місцях) Метод сортування – вибір.
15	Дана послідовність цілих чисел $a_1, a_2, \ldots, a_{30}$. Розташувати додатні елементи послідовності, які кратні 5, за спаданням (інші елементи залишаються на своїх місцях). Метод сортування – вибір.

Короткі теоретичні відомості

Найбільш простий зі способів пошуку даних – лінійний пошук. Даний алгоритм порівнює кожний елемент масиву із ключем, наданим для пошуку. Алгоритм лінійного пошуку відмінно працює тільки для невеликих або неупорядкованих масивів і є абсолютно надійним.

Якщо масив містить упорядковану послідовність даних, то більш ефективним буде двійковий (бінарний) пошук.

Припустимо, що змінні *Lb* і *Rb* містять, відповідно, ліву й праву границі відрізка масиву, де перебуває потрібний елемент. Пошук завжди будемо починати з аналізу середнього елементу *M* відрізка масиву. Якщо шукане значення менше *M*, переходимо до пошуку в лівій половині відрізка, де всі елементи менші за елемент, який тільки що перевірили. Інакше кажучи, значенням *Rb* стає (*M*-1) і на наступній ітерації робота ведеться з половиною масиву. Таким чином, у результаті кожної перевірки вдвічі звужується область пошуку.

Двійковий пошук – дуже потужний метод, наприклад, якщо довжина масиву рівна 1023, тоді після першого порівняння проміжок звужується до 511 елементів, а після другого – до 255, тобто для пошуку в масиві з 1023 елементів досить 10 порівнянь.

Ідея методу бульбашкового сортування полягає в наступному: крок сортування полягає в проході знизу вгору по масиву. По дорозі проглядаються пари сусідніх елементів. Якщо елементи деякої пари знаходяться в неправильному порядку, то вони міняються місцями. Після першого проходу по масиву "вгорі" виявляється самий "легкий" елемент. Наступний прохід робиться до другого зверху елемента, таким чином другий за величиною елемент піднімається на правильну позицію. Перегляд елементів робиться по все зменшуючій нижній частині масиву до тих пір, поки в ній не залишиться тільки один елемент. На цьому сортування закінчується, так як послідовність впорядкована за зростанням.

Основні принципи методу: середня кількість порівнянь та перестановок мають квадратичний порядок зростання, звідси можна зробити висновок, що алгоритм бульбашки дуже повільний і малоефективний. Проте, у нього є величезний плюс: він простий і його можна по-всякому покращувати.

Якщо на якомусь із проходів не відбулося жодного обміну, це означає, що всі пари розташовані в правильному порядку, тобто масив вже відсортований. Таким чином перший крок оптимізації полягає в запам'ятовуванні, чи проводився при даному проході будь-який обмін. Якщо ні – алгоритм закінчує роботу.

Ідея методу сортування вибором полягає в тому, щоб створювати відсортовану послідовність шляхом приєднання до неї одного за одним елементів в правильному порядку. Алгоритм складається з n послідовних кроків, починаючи від нульового і закінчуючи (n-1). На i-му кроці вибираємо найменший з елементів $a[i]$ ... $a[n]$ і міняємо його місцями з $a[i]$. Незалежно від номеру поточного кроку i, послідовність $a[0]$ ... $a[i]$ є впорядкованою. Таким чином, на етапі (n-1) вся послідовність, крім $a[n]$ елементу виявляється відсортована,

а $a[n]$ стоїть на останньому місці по праву: все менші елементи вже пішли вліво.

Основні принципи методу: для знаходження найменшого елемента з $n+1$ розглядаємих елементів алгоритм робить n порівнянь. Оскільки кількість перестановок завжди буде меншою за кількість порівнянь, час сортування зростає щодо кількості елементів. Крім того, алгоритм не використовує додаткову пам'ять: всі операції відбуваються "на місці".

Сортування цим методом можна використовувати для масивів, що мають невеликі розміри.

Сортування простими вставками в чомусь схоже на методи, наведені раніше. Аналогічним чином робляться проходи по частині масиву, і аналогічним же чином на його початку "виростає" відсортована послідовність.

Однак в сортуванні бульбашкою або вибором можна було чітко заявити, що на i-му кроці елементи $a[0] \ldots a[i]$ стоять на правильних місцях і нікуди вже не перемістяться. В цьому ж сортуванні подібне твердження буде більш слабким: послідовність $a[0] \ldots a[i]$ впорядкована. При цьому по ходу алгоритму в неї будуть вставлятися нові елементи.

Розглянемо дії алгоритму на i-му кроці. Послідовність до цього моменту розділена на дві частини: впорядковану $a[0] \ldots a[i]$ та невпорядковану $a[i+1] \ldots a[n]$. На наступному, $(i+1)$-му кожному кроці алгоритму беремо $a[i+1]$-й елемент і вставляємо на потрібне місце у відсортовану частину масиву. Пошук відповідного місця для чергового елементу вхідної послідовності здійснюється шляхом послідовних порівнянь з елементом, що стоять перед ним. Залежно від результату порівняння елемент або залишається на поточному місці (вставка завершена), або вони міняються місцями і процес повторюється.

Гарним показником сортування є дуже природна поведінка: майже відсортований масив буде упорядкований дуже швидко. Це, вкупі зі стійкістю алгоритму, робить метод хорошим вибором у відповідних ситуаціях.

Приклад виконання роботи

Завдання 1.1. Сформулювати математичний запис фрагмента програми та обчислити значення змінної `x` після його виконання, де `size=22` – розмір масиву, `N=31` – варіант за списком групи.

```
for (int i=0; i<size-1; i++) {
      if(a[i]%2==0)
      {int nmin = i;
        for (int j=i+1; j<size; j++){
            if (a[j]<a[nmin]&& (a[j]%2==0)) nmin = j;
        }
        if (i != nmin) {
            int t = a[nmin];
            a[nmin] = a[i];
            a[i] = t;
        }
    }
      }
x=a[3];
```

Розв'язання

Даний фрагмент програми виконує сортування парних елементів масиву за зростанням. Реалізація сортування виконана методом вибору, зміна `x=-38`.

Завдання 1.2. Дана послідовність цілих чисел $a_1, a_2, ..., a_{35}$. Розташувати елементи послідовності, кратні 5 за зростанням (інші елементи залишаються на своїх місцях). Сортування виконати методом вибору.

Розв'язання

1. Постановка задачі

Дана послідовність цілих чисел a_1, a_2, ..., a_{35}. Відсортувати за зростанням тільки елементи, які кратні 5.

2. Алгоритм розв'язання задачі.

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ввести елементи одновимірного масиву *a*.

Дія 2. Відсортувати елементи, які кратні 5.

Дія 3. Вивести упорядкований масив на екран.

3. Текст програми

```
#include <iostream>
#include <windows.h>
using namespace std;
void chooseSort(int a[], int size){
for (int i=0; i<size-1; i++) {
      if(a[i]%5==0)
      {int nmin = i;
        for (int j=i+1; j<size; j++){
             if (a[j]<a[nmin]&&(a[j]%5==0)) nmin = j;
        }
        if (i != nmin) {
            int t = a[nmin];
            a[nmin] = a[i];
            a[i] = t;
        }
    }
      }
}
void printArr(int a[], int size){
    for(int i=0; i<size; i++) {
        cout << a[i] << " ";
    }
    cout << "\n";
}
```

```
int main(){
   SetConsoleCP(1251);SetConsoleOutputCP(1251);
   int p[22], arr[22],n=22;
   p[0]=31;
   for(int i=0;i<n-1;i++)
   p[i+1]=(p[i]*11 + 7) % 100;
      for(int i=0;i<n;i++)
        arr[i]=p[i]-50;
  cout<<"До сортування:\n";
    printArr(arr, n);
    chooseSort(arr, n);
  cout<<"Після сортування:\n";
    printArr(arr, n);
    return 0;
}
```

4. Результати роботи програми

```
До сортування
-19 -2 -15 42 -31 -34 33 -30 -23 -46 1 18 5 -38 -11 -
14 -47 -10 -3 -26 21 38
Після сортування
-19 -2 -30 42 -31 -34 33 -15 -23 -46 1 18 -10 -38 -11
-14 -47 5 -3 -26 21 38
```

Контрольні питання

1. Який основний принцип роботи алгоритму лінійного пошуку?

2. До якого масиву можна застосовувати алгоритм бінарного пошуку?

3. Який основний принцип роботи алгоритму бінарного пошуку?

4. Який основний принцип роботи алгоритму бульбашкового сортування?

5. Як працює алгоритм сортування вибором?

6. Як працює алгоритм сортування простими вставками?

Робота № 2

Розробка та реалізація програм для роботи з вказівниками та одновимірними динамічними масивами

Мета роботи: оволодіння навичками складання програм для роботи з вказівниками та одновимірними динамічними масивами та виконання їх в NetBeans IDE.

Завдання

Завдання 2.1. Нехай є наступний фрагмент програми, поданий у варіантах (див. табл. 2.1). Поясніть, як зміниться масив після його виконання, при наступному початку програми (Вказівка: замість N підставити номер варіанта за списком групи).

```
#include <iostream>
#include<cmath>
using namespace std;
int* form(int&n) {  n=10+N%10;
     int*a=new int[n];
     *a=N;
     for(int i=1;i<n;i++)
          *(a+i)=*(a+i-1)+1;
     return a;
}
int main(){
```

Таблиця 2.1 – Варіанти завдання 2.1

№	Фрагмент програми

1–5	`int *a, n;  a=form(n);`
	`int **p=new int *[n];`
	`for (int i=0;i<n;i++)`
	`{ p[i]=(a+i);`
	`*p[i]=*p[i]+N;`
	`}`

Продовж. табл. 2.1

№	Фрагмент програми

6–10	`int *a, n,i,*p;`
	`a=form(n);`
	`for (p = &a[0], i = 0 ; i <n; i++ )`
	`{`
	`*(p+i) -=N*N;`
	`}`

11–15	`int *a, n,i,*p;`
	`a=form(n);`
	`for ( p = a, i = 0; p+i < a+n; i++ ){`
	`*(p+i) *=*(p+i);`
	`}`

16-20	`int *a, n,i,*p;`
	`a=form(n);`
	`for ( p = a+n, i=1; i <=n; i++ )`
	`{`
	`*(p-i) *= (int)sqrt(N);`
	`}`

21–25	`int *a, n,i,*aa; a=form(n);`
	`int **p=new int *[n];`
	`aa=a+n-1;`
	`for (i=0;aa-a>=0;aa--,i++)`
	`{ *(p+i)=aa;`
	`**(p+i)+=*(a+i);`
	`}`

26–30	`int *a, n,i,t=0,*p; a=form(n);`
	`int *aa=new int[n]; p=aa;`
	`for (i=n-1;i>=0;i--)`
	`{*p++=*(a+i); t+=*(a+i);}`
	`t=t/n;`
	`for (i=0;i<n;i++)`
	`*(a+i)=*(aa+i)+t;`
	`}`

Завдання 2.2. Скласти програму, що заповнює динамічний одновимірний масив за допомогою випадкових чисел, потім змінює масив, згідно завдань, які наведені в таблиці 2.2, та виконати її в NetBeans IDE.

Примітка:

a. Вивести масив до та після модифікації.

b. При звертанні до елементів масиву використовувати вказівники.

Таблиця 2.2 – Варіанти завдання 2.2

№	Завдання
1.	Видалити всі від'ємні двозначні елементи масиву.
2.	Видалити елемент із заданим ключем (значенням) і елемент із заданим номером.
3.	Видалити всі елементи, розташовані між першим парним елементом і елементом p, де p – це ціла частина середнього арифметичного елементів масиву.
4.	Видалити N елементів, починаючи з номера K, де N і K вводяться з клавіатури.
5.	Видалити всі парні елементи.
6.	Видалити всі елементи з парними індексами.
7.	Видалити всі непарні елементи.
8.	Видалити всі елементи з непарними індексами.
9.	Додати K простих чисел на початку масиву, де K вводиться з клавіатури.
10.	Додати K простих чисел в кінець масиву, де K вводиться з клавіатури.
11.	Додати K чисел Фібоначчі на початку масиву, де K вводиться з клавіатури.
12.	Додати K чисел Фібоначчі в кінець масиву, де K вводиться з клавіатури.
13.	Додати K елементів, починаючи з номера N, де N і K вводяться з клавіатури.
14.	Додати після кожного від'ємного елемента його абсолютне значення.
15.	Додати після кожного парного елемента, елемент зі значенням 0.

Короткі теоретичні відомості

Вказівник – це змінна, яка містить адресу іншої змінної. Вказівники дуже широко використовуються в мові C++. Іноді вони дають єдину можливість виразити потрібну дію, та зазвичай ведуть до більш компактних і ефективних програм.

Так як вказівник містить адресу об'єкта, це дає можливість "непрямого" доступу до цього об'єкта через вказівник. Припустимо, що `x` – змінна, наприклад, типу `int`, а `px` – вказівник, створений ще не зазначеним способом. Унарна операція `&` видає адресу об'єкта, так що оператор `px = &x;` присвоює адресу `x` змінній `px`; кажуть, що `px` "вказує" на `x`. Операція `&` застосовується тільки до змінних та елементів масиву. Не можна отримати адресу регістрової змінної.

Унарна операція `*` розглядає свій операнд як адресу кінцевої мети і звертається за цією адресою, щоб витягти вміст. Отже, якщо `y` має тип `int`, то `y=*px;` присвоює `y` вміст того, на що вказує `px`. Так послідовність `px=&x; y=*px;` присвоює `y` те ж саме значення, що і оператор `y=x;`

Опис вказівника `int *px;` є новим і має розглядатися як мнемонічний; такий запис, треба читати: вказівник `px` вказує на тип `int`. Це означає, що якщо `px` з'являється в контексті `*px`, то це еквівалентно змінній типу `int`.

Вказівники можуть входити у вираз. Наприклад, якщо `px` вказує на ціле `x`, то `*px` може з'являтися в будь-якому контексті, де може зустрітися `x`. У виразах виду `y = *px + 1;` унарні операції `*` та `&` пов'язані зі своїм операндом міцніше, ніж арифметичні операції, так що такий вислів бере значення, на яке вказує `px`, додає `1` і присвоює результат змінній `y`.

Вказівники є змінними, з ними можна поводитися, як і з іншими змінними. Якщо `py` – інший вказівник на змінну типу `int`, то `py = px;` копіює вміст `px` в `py`, в результаті чого `py` вказує на те ж, що і `px`.

Використання вказівників як альтернативний спосіб доступу до змінних приховує в собі небезпеку – якщо була змінена адреса, що зберігається у вказівнику, то цей вказівник більше не посилається на потрібне значення.

Мова C++ пропонує альтернативу для більш безпечного доступу до змінних через вказівники. Оголосивши посилальну змінну, можна створити об'єкт, який, як і вказівник, буде посилатися на інше значення, але, на відміну від вказівника, постійно прив'язаний до цього значення. Таким чином, посилання на значення завжди посилається на це значення.

Синтаксис оголошення посилання:

```
<ім'я типу> & <ім'я посилання> = <вираз>;
```

або

```
<ім'я типу> & <ім'я посилання> (<вираз>);
```

Одного разу ініціалізувавши посилання йому не можна присвоювати інше значення. На відміну від вказівників, які можуть бути оголошені неініціалізовані або встановлені в нуль (`NULL`), посилання завжди посилаються на об'єкт. Для посилань обов'язкова ініціалізація при створенні і не існує аналога нульового вказівника.

Посилання не можна ініціалізувати в наступних випадках:

– при використанні в якості параметрів функції;

– при використанні в якості типу значення, що повертає функція;

– в оголошеннях класів.

Не існує операторів, що безпосередньо виконують дії над посиланнями.

Посилальні змінні використовуються досить рідко, значно зручніше використовувати саму змінну, ніж посилання на неї. Більш широке застосування посилання мають в якості параметрів функції. Посилання особливо корисні у функціях,

які повертають декілька об'єктів (значень). Посилання та вказівники в якості параметрів функцій тісно пов'язані.

В C++ передача аргументів функцій здійснюється "за значенням", тобто змінні, що передаються у функцію, не можуть поміняти своє значення. Якщо треба змінити аргументи функції, тоді в якості аргументів треба використовувати посилання: `void func (int &a,int &b);`

У мові C++ існує взаємозв'язок між вказівниками та масивами. Будь-яку операцію, яку можна виконати за допомогою індексів масиву, можна виконати й за допомогою вказівників.

Опис `int a[10];` визначає масив розміру `10`. Запис `a[i]` відповідає елементу масиву через `i` позицій від початку. Якщо `pa` – вказівник цілого, описаний як `int *pa;` тоді присвоювання `pa = &a[0]` приводить до того, що `pa` вказує на нульовий елемент масиву `a`. Це означає, що `pa` містить адресу елемента `a[0]`. Присвоювання `x = *pa` буде копіювати вміст `a[0]` в `x`.

Якщо `pa` вказує на певний елемент масиву `a`, тоді `pa+1` вказує на наступний елемент, а `pa-i` вказує на елемент, що стоїть на `i` позицій до елемента, на який вказує `pa`, `pa+i` вказує на елемент, що стоїть на `i` позицій після. Таким чином, якщо `pa` вказує на `a[0]`, тоді `*(pa+1)` посилається на вміст `a[1]`, `pa+i` – адреса `a[i]`, а `*(pa+i)` – вміст `a[i]`.

Ці зауваження справедливі незалежно від типу змінних у масиві `a`. Суть визначення "додавання 1 до вказівника", а також його поширення на всю арифметику вказівників, полягає в тому, що збільшення масштабується розміром пам'яті, що займає об'єкт, на який вказує вказівник. Таким чином, `i` в `pa+i` перед збільшенням множиться на розмір об'єктів, на які вказує `pa`.

Посилання на `a[i]` можна записати у вигляді `*(a+i)`. Ці дві форми еквівалентні. Якщо застосувати операцію `&` до

обох частин такого співвідношення еквівалентності, то ми одержимо, що `&a[i]` та `a+i` ідентичні: `a+i` – адреса `i`-го елемента від початку `a`. З іншого боку, якщо `pa` є вказівником, то у виразах його можна використовувати з індексом: `pa[i]` ідентично `*(pa+i)`.

Є одна відмінність між іменем масиву та вказівником: вказівник є змінною, так що операції `pa=a` та `pa++` мають сенс. Але ім'я масиву є константою, а не змінною: конструкції типу `a=pa` або `a++`, або `p=&a` будуть помилковими.

Динамічна пам'ять або пам'ять вільного зберігання відрізняється від статичної тим, що програма повинна явно запросити пам'ять для елементів, збережених у цій області, а потім звільнити пам'ять, якщо вона більше не потрібна.

За допомогою операції виділення пам'яті можна виділяти пам'ять динамічно, тобто на етапі виконання програми:

```
вказівник_на_тип_= new ім'я_типу (ініціалізатор);
```

Ініціалізатор – це необов'язковий вираз, який може використовуватися для всіх типів, крім масивів.

При виконанні оператору `int *ip = new int;` створюються 2 об'єкта: динамічний безіменний об'єкт та вказівник на нього з іменем `ip`, значенням якого є адреса динамічного об'єкта. Якщо вказівнику `ip` присвоїти інше значення, то можна втратити доступ до динамічного об'єкта. У результаті динамічний об'єкт як і раніше буде існувати, але звернутися до нього вже не можна. Такі об'єкти називаються сміттям.

При виділенні пам'яті об'єкт можна ініціалізувати: `int *ip = new int(3);` Можна динамічно розподілити пам'ять і під масив: `double *mas = new double [50];` Далі із цією динамічно виділеною пам'яттю можна працювати як зі звичайним масивом.

У випадку успішного завершення операція `new` повертає вказівник зі значенням, відмінним від нуля. Результат операції, рівний `0`, тобто нульовому вказівнику `NULL`, говорить про те, що не знайдений неперервний вільний фрагмент пам'яті потрібного розміру.

Операція звільнення пам'яті `delete` – звільняє ділянку пам'яті для подальшого використання в програмі, яка раніше була виділена операцією `new: delete ip;` – видаляє динамічний об'єкт типу `int`, який був створений за допомогою команди `ip = new int;` або `delete [ ] mas;` – видаляє динамічний масив який був створений за допомогою команди `double *mas = new double[50];`

Безпечно застосовувати операцію до вказівника `NULL`. Результат же повторного застосування операції `delete` до одного й того ж вказівника не визначений. Звичайно відбувається помилка, що призводить до зациклення. Щоб уникнути подібних помилок, слід виконувати перевірку `if (ip)` перед застосуванням операції `delete`.

Приклад виконання роботи

Завдання 2.1. Пояснити, що виконує наступний фрагмента програми, де `N=32`.

```
int *a, n,*aa,t; a=form(n);
for (aa=a; a+n-1-aa>=0; aa+=2){
    t=*aa; *aa=*(aa+1); *(aa+1)=t; }
```

Розв'язання

Програма реалізує спочатку формування лінійного масиву $\{a_i\}$, де $i = \overline{1,12}$

```
32 33 34 35 36 37 38 39 40 41 42 43
```

Наданий фрагмент програми переставляє місцями елементи з парними та непарними індексами.

```
33 32 35 34 37 36 39 38 41 40 43 42
```

Завдання 2.2. Видалити з масиву всі елементи, що співпадають з першим елементом. При складанні програми використовувати динамічне розподілення пам'яті та вказівники, виконати її в NetBeans IDE.

Розв'язання

1. Постановка задачі

Скласти програму видалення з масиву всіх елементів, що співпадають з першим елементом на мові С++.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Виділити динамічну пам'ять для елементів масиву.

Дія 2. Заповнити масив випадковими числами.

Дія 3. Вивести елементи масиву на екран.

Дія 4. Визначити кількість елементів, що відрізняються від першого.

Дія 5. Виділити динамічну пам'ять для нового масиву.

Дія 6. У новий масив записати тільки ті елементи, що відрізняються від першого.

Дія 7. Звільнити пам'ять, що була виділена під старий масив.

Дія 8. Замінити значення вказівки

3. Текст програми

```
#include <iostream>
#include <stdlib.h>
#include <windows.h>
using namespace std;
int* form(int&n){
```

```
     cout<<"\nВведіть n=";
     cin>>n;
     int*a=new int[n];
     for(int i=0;i<n;i++)
          *(a+i)=rand()%100;
     return a;
}
void print(int*a,int n){
     for(int i=0;i<n;i++)
          cout<<*(a+i)<<" ";
     cout<<"\n";
}
int count0(int*a,int n){
int k=0,i=0;
for(;i<n;i++)
                 if(*(a+i)!=*(a+0)) k++;
return k;
}
int*dell(int *a,int&n){
     int newsize=count0(a,n);
int*b=new int [newsize];
for(int j=0, i=0;i<n;i++)
          if(*(a+i)!=*(a+0))
          {
               *(b+j)=*(a+i);j++;
          }
    n=newsize;
    delete [] a;
    return b;
}
int main(){SetConsoleCP(1251);SetConsoleOutputCP(1251);
     int *a, n;    a=form(n);
    cout<<"До видалення:\n";print(a,n);
     a=dell(a,n);
     cout<<"Після видалення:\n";print(a,n);
    delete [] a;
     return 0;
}
```

4. Результати роботи програми

```
Введіть n=10
До видалення:
41 67 34 0 69 24 78 58 62 64
Після видалення:
67 34 0 69 24 78 58 62 64
```

Контрольні питання

1. Що таке вказівник?
2. Як працює унарна операція `&`?
3. Як працює унарна операція `*`?
4. Що таке посилальна змінна?
5. Де найбільш часто використовуються посилальні змінні?
6. Як можна звернутися до певного елемента масиву через індекс?
7. Як можна звернутися до певного елемента масиву через вказівник?
8. Що таке динамічна пам'ять?
9. Як працює оператор `new`?
10. Як працює оператор `delete`?

Робота № 3

Розробка та реалізація програм з використанням багатовимірних динамічних масивів

Мета роботи: оволодіння навичками складання програм з використанням багатовимірних динамічних масивів та виконання їх в NetBeans IDE.

Завдання

Завдання 3.1. Нехай є наступний фрагмент програми, поданий у варіантах (див. табл. 3.1). Поясніть, яку задачу реалізує вказаний фрагмент, при наступному початку програми (Вказівка: замість `N` підставити номер варіанта за списком групи).

```
#include <iostream>
#include <stdlib.h>
#include <iomanip>
const int N=32;
using namespace std;
int** form_matr(int n){
  int **matr=new int*[n];
  for(int i=0;i<n;i++) matr[i]=new int [n];
  return matr;}
void zapolnen (int ** matr,int n){
int p=N;
     for(int i=0;i<n;i++)
     for(int j=0;j<n;j++){     matr[i][j]=p++ ;
}}
void printArray (int ** matr,int n){
     for(int i=0;i<n;i++){
          for(int j=0;j<n;j++)
               cout<<setw(5)<<matr[i][j];
          cout<<"\n";    }}
int main(){
```

```
int n;
n=10+N%10;
int **matr=form_matr(n);
zapolnen(matr,n);
printArray(matr,n);
```

Таблиця 3.1 – Варіанти завдання 3.1

№	Фрагмент програми

1–5	`int x=0;`
	`for(int i=0;i<n;i++)`
	`for(int j=i;j<=n-i-1; j++){`
	`x+=matr[i][j] ;`
	`}`

6–10	`int t;`
	`for(int i=0;i<n;i++)`
	`for(int j=i;j<=n-i-1; j++) {`
	`t=matr[i][j] ;`
	`matr[i][j]=matr[n-i-1][j];`
	`matr[n-i-1][j]=t;}`

11–15	`int x=matr[1][n-1], m=0;`
	`for(int i=0;i<n;i++){`
	`for(int j=n-1-i;j<n;j++)`
	`if (matr[i][j]<x){`
	`x=matr[i][j]; m=i;}}`

16–20	`int x=matr[1][n-1], m=0;`
	`for(int i=0;i<n;i++){`
	`for(int j=0;j<=n-1-i; j++)`
	`if (matr[i][j]>x){`
	`x=matr[i][j]; m=i;}}`

21–25	`int t;`
	`for(int j=0;j<n/2;j++)`
	`for(int i=j;i<=n-j-1; i++){`
	`t=matr[i][n-1-j] ;`
	`matr[i][n-1-j]= matr[i][j];`
	`matr[i][j]=t;}`

26–30	`double x=1;`
	`for(int i=0;i<n;i++){`
	`for(int j=0;j<=i;j++)`
	`x*=matr[i][j];`
	`}`

Завдання 3.2. Розробити програму, яка виконує обчислення над двовимірними динамічними масивами, при написанні програми користуватися принципами структурного програмування (описати такі функції: що створює масив, що заповнює його випадковими числами, що друкує, що реалізує завдання з таблиці 3.2).

Таблиця 3.2 – Варіанти завдання 3.2

№	Завдання
1	Дана дійсна матриця розміром 7 * 8. Знайти найбільший елемент матриці. Поміняти рядок, що містить найбільший елемент з першим рядком матриці.
2	Дана дійсна матриця розміром 7 * 8. Знайти найбільший елемент матриці. Поміняти стовпчик, що містить найбільший елемент з першим стовпцем матриці.
3	Скласти програму заміни всіх від'ємних елементів матриці **A** (9,9) на 0, якщо сума мінімального і максимального елементів цієї матриці виявиться менше *P*, де *P* вводиться з клавіатури.
4	Скласти програму знаходження максимального елемента в кожному стовпчику матриці **A** (15,15).
5	Скласти програму знаходження мінімального додатнього елементу в кожному стовпці матриці **A** (11,11).
6	Скласти програму знаходження кількості рядків матриці **A** (10,10), сума елементів яких від'ємна.
7	Скласти програму знаходження кількості рядків матриці **A** (8, 8), кількість від'ємних елементів в яких більше за *P*, де *P* вводиться з клавіатури.
8	Скласти програму формування вектора **B** (16), якщо b_i – сума мінімального і максимального елементів *i*-го рядку матриці **A** (16,5).
9	Скласти програму заміни всіх від'ємних елементів матриці **A** (12,12) на елемент цієї матриці, що має максимальне значення.
10	Є двовимірний масив цілих чисел розмірності 6 * 15. Знайти номер рядка, для якого середньоарифметичне значення елементів максимально.
11	В двовимірному масиві цілих чисел розмірності 9 * 9 поміняти місцями рядок і стовпчик, номер якого вводиться користувачем.

Продовж. табл. 3.2

№	Завдання
12	Дано масив **C** (14,14). Визначити кількість "особливих" елементів масиву, вважаючи елемент "особливим", якщо він більше суми інших елементів свого стовпчика. Надрукувати індекси "особливих" елементів.
13	Дано масив **C** (11,11). Визначити кількість "особливих" елементів масиву, вважаючи елемент "особливим" якщо в його рядку зліва від нього знаходяться елементи менші за нього, а праворуч більші.
14	Знайти мінімальний елемент серед максимальних елементів рядків двовимірного масиву цілих чисел. Визначити номер рядка і стовпчика такого елементу.
15	Видалити стовпець двовимірного масиву цілих чисел, в якому знаходиться максимальний елемент цього масиву.
16	Дано масив **C** (10,10). Знайти всі неповторювані елементи двовимірного масиву цілих чисел.
17	Дано масив **C** (7,10). Видалити стовпчики, розташовані між стовпчиками з мінімальним та максимальним елементами.
18	Дано масив **C** (9,11). Парні рядки масиву зсунути циклічно на *K* елементів вправо, де *K* вводиться з клавіатури.
19	Дано масив **C** (9,12). Непарні рядки масиву зсунути циклічно на *K* елементів вліво, де *K* вводиться з клавіатури.
20	Створіть двовимірний масив цілих чисел **C** (11,9). Видаліть з нього рядок і стовпець, на перетині яких розташований мінімальний елемент.

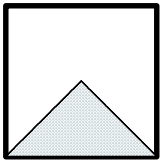
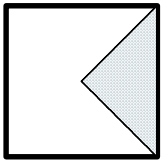
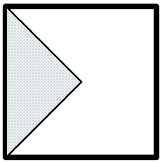
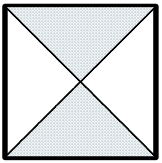
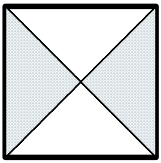
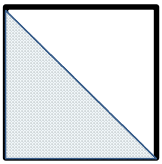
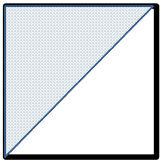
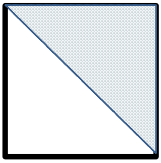
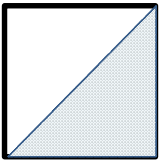
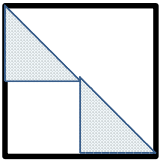
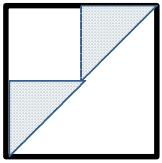
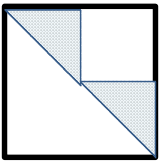
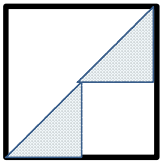
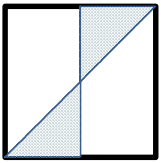
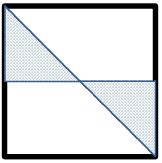
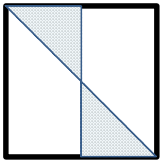
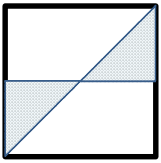
Завдання 3.3. Скласти програму, яка модифікує двовимірні динамічні масиви. При написанні програми користуватися принципами структурного програмування (описати такі функції: створити масив, сформувати масив, друкувати на екрані, модифікувати масив, згідно завдання з таблиці 3.3, звільнити пам'ять).

Таблиця 3.3 – Варіанти завдання 3.3

№	Завдання
1	Додати рядок з вказаним номером K
2	Додати рядок з початку матриці
3	Додати стовпчик з початку матриці
4	Додати K рядків з початку матриці
5	Додати K стовпців з початку матриці
6	Видалити рядок з номером K
7	Видалити стовпець з номером K
8	Видалити рядки, починаючи з рядка K_1 і до рядка K_2
9	Видалити стовпці, починаючи з стовпця K_1 і до стовпця K_2
10	Видалити всі парні рядки
11	Видалити всі парні стовпці
12	Видалити всі рядки, в яких є хоча б один нульовий елемент
13	Видалити всі стовпці, в яких є хоча б один нульовий елемент
14	Видалити рядок, в якому знаходиться найбільший елемент матриці
15	Додати рядки після кожного парного рядку матриці
16	Додати стовпці після кожного парного стовпця матриці
17	Додати K рядків, починаючи з рядка з номером N
18	Додати K стовпців, починаючи зі стовпця з номером N
19	Додати рядок після рядка, що містить найбільший елемент
20	Додати стовпчик після стовпчика, що містить найбільший елемент

Завдання 3.4. Розробити програму, яка заповнює двовимірний масив наступним чином: елементи, що належать заштрихованій області, генеруються випадковим чином; всі інші дорівнюють `N`, де `N` – номер варіанта. Упорядкувати всі випадкові елементи масиву, які належать заштрихованій області. Парні варіанти упорядковуються за зростанням, непарні – за спаданням. При написанні програми користуватися принципами структурного програмування.

Таблиця 3.4 – Варіанти завдання 3.4

№	Рисунок	№	Рисунок	№	Рисунок
1		2		3	
4		5		6	
7		8		9	
10		11		12	
13		14		15	
16		17		18	

Короткі теоретичні відомості

Окремим випадком багатовимірного масиву є двовимірний масив, або матриця. Двовимірний масив являє собою сукупність рядків і стовпців, на перетині яких перебуває конкретне значення. Для оголошення двовимірного масиву необхідно вказати кількість рядків і стовпців. При цьому діють ті ж правила, що й при оголошенні одновимірного масиву:

```
тип_даних ім'я_масиву [число_рядків][число_стовпців];
```

Незважаючи на те, що ми уявляємо двовимірний масив у вигляді матриці, у пам'яті будь-який двовимірний масив розташовується по рядкам: спочатку нульовий рядок, потім перший і так далі. Про це слід пам'ятати, тому що вихід за межі масиву може викликати некоректну роботу програми, при цьому компілятор не повідомлює про помилку.

Звернення до певного елементу масиву здійснюється за номером рядка та номером стовпця, наприклад: `array [2][1]`.

Багатовимірний масив в C++ по своїй суті одновимірний. Операції `new` та `delete` дозволяють створювати й видаляти динамічні масиви, підтримуючи при цьому ілюзію довільної розмірності. Діяльність по організації динамічного масиву вимагає додаткової уваги, однак характеристики масиву (операнди операції `new`) можуть не бути константними виразами. Це дозволяє створювати багатовимірні динамічні масиви довільної конфігурації. Більш докладно робота з операціями `new` та `delete` була розглянута у роботі №2.

Організація двовимірного динамічного масиву виконується у два етапи: спочатку створюється одновимірний масив вказівників, а потім кожному елементу цього масиву присвоюється адреса одновимірного масиву:

```
int size_row = 5, size_col = 5;
int **pArr = new int*[ size_row];
for (int i = 0; i < size_row; i++)
pArr[i] = new int[size_col];
```

Знищення двовимірного масиву відбувається у зворотній послідовності:

```
for (int i = 0; i < size_row; i++)
          delete[]pArr[i];
      delete[]pArr;
```

Приклад виконання роботи

Завдання 3.1. Пояснити, що виконує наступний фрагмент програми, де `N=32`.

```
int x=0;
for(int i=0;i<n;i++){
    for(int j=0;j<=n-1-i;j++)
     x+=matr[i][j];
```

Розв'язання

Цей фрагмент програми реалізує обчислення суми елементів, що розташовані на та над другою діагоналлю матриці. Після виконання цього фрагмента `x = 208`.

Завдання 3.2. Скласти програму. Видалити з двовимірного динамічного масиву рядок, що містить максимальний елемент.

Розв'язання

1. Постановка задачі

Скласти програму, що видаляє з двовимірного динамічного масиву рядок, що містить максимальний елемент, на мові С++.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Виділити динамічну пам'ять для елементів масиву.

Дія 2. Заповнити масив випадковими числами.

Дія 3. Вивести елементи масиву на екран.

Дія 4. Визначити індекс рядка, в якому розташований максимальний елемент.

Дія 5. Виділити динамічну пам'ять для нового масиву.

Дія 6. У новий масив записати тільки ті рядки, що відрізняються від знайденого індексу.

Дія 7. Звільнити пам'ять, що була виділена під старий масив.

Дія 8. Замінити значення вказівки

3. Текст програми

```
#include <iostream>
#include <stdlib.h>
#include <iomanip>
using namespace std;
/*виділення динамічної пам'яті під двовимірний дина-
мічний масив*/
int** form_matr(int n,int m){
int **matr=new int*[n];
for(int i=0;i<n;i++)
     matr[i]=new int [m];
  return matr;//повертаємо вказівку на масив вказівок
}
void zapolnen (int ** matr,int n,int m){
//заповнення матриці
     for(int i=0;i<n;i++)
     for(int j=0;j<m;j++)
          matr[i][j]=rand()%100;
}
void printArray (int ** matr,int n,int m){
     for(int i=0;i<n;i++){
          for(int j=0;j<m;j++)
               cout<<setw(5)<<matr[i][j];
          cout<<"\n";}
}
void dell(int ** matr,int n,int m){//звільнення пам'яті
     for(int i=0;i<n;i++)
```

```
          delete [] matr[i];
     delete[]matr;
}
//пошук індексу рядка з максимальним елементом
int maxIndex (int ** matr,int n,int m){
int max=matr[0][0],imax=0;
     for(int i=0;i<n;i++)
      for(int j=0;j<m;j++)
          if (max<matr[i][j]) {   max=matr[i][j];
                                  imax=i;}
     return imax;
     }
int ** deleteMax (int ** matr,int &n,int m){
     int**temp=form_matr(n-1,m);
          //заповнення нової матриці
     int t=0,k=maxIndex(matr,n,m);
     for(int i=0;i<n;i++)
          if(i!=k)
          {
               for(int j=0;j<m;j++)
                    temp[t][j]=matr[i][j];
               t++;
          }
     dell(matr, n, m);
    n--;
    return temp;
}
int main()
{
     int n,m;//розмір матриці
     cout<<"\nEnter n=";
     cin>>n;//рядки
     cout<<"\nEnter m=";
     cin>>m;//стовпці
     int **matr=form_matr(n,m);
     zapolnen(matr,n,m);
  cout<<"old array:\n";
     printArray(matr,n,m);
```

```
     matr=deleteMax(matr,n,m);
     cout<<"new array:\n";
    printArray(matr,n,m);
    dell(matr, n, m);
     return 0;
}
```

4. Результати роботи програми

```
Enter n=5
Enter m=5
old array:
   41   67   34    0   69
   24   78   58   62   64
    5   45   81   27   61
   91   95   42   27   36
   91    4    2   53   92
new array:
   41   67   34    0   69
   24   78   58   62   64
    5   45   81   27   61
   91    4    2   53   92
```

Контрольні питання

1. Як оголосити двовимірний масив?
2. Які існують варіанти ініціалізації двовимірного масиву?
3. Продемонструйте взаємозв'язок між посиланнями та масивами.
4. Що таке динамічна пам'ять?
5. Яка існує особливість динамічного виділення пам'яті у випадку двовимірного масиву?
6. Яка існує особливість звільнення пам'яті у випадку двовимірного масиву?

Робота № 4

Розробка та реалізація програм для роботи зі структурами

Мета роботи: оволодіння навичками складання програм для роботи зі структурами та виконання їх в NetBeans IDE.

Завдання

Завдання 4.1. Нехай є наступний фрагмент програми, поданий у варіантах (див. табл. 4.1). Поясніть, яку задачу реалізує вказаний фрагмент, при наступному початку програми.

```
#include <iostream>
#include <cstring>
#include<cctype>
using namespace std;
const int N=4;
struct myType
{    char fff[20];    char aaa[20];
    int b;    double c;};
void printStruct(myType st[],int n){
for (int i=0;i<n;i++){
  cout << " fff: " << st[i].fff << "  aaa: "<< st[i].aaa ;
  cout << "  b : " << st[i].b<<"  c : " << st[i].c << endl;}}
int main(){
     myType st[]={{"assdss","ssdhshf",4,4.5},
                  {"jherhhgb","xczvfdg",9,2.4},
                  {"xcvfd","hhgeryt",5,4.1},
                  {"jrhjdfd","tyyyyy",7,3.5}   };
printStruct(st,N);
```

Таблиця 4.1 – Варіанти завдання 4.1

№	Фрагмент програми

1–5	`double k=0; char c='d';`
	`for (int i=0; i<N; i++){`
	`if (strchr(st[i].fff,c)) k++;`
	`}`

6–10	`char t[20];`
	`for (int i=0; i< N; i++) {`
	`for (int j= N-1; j>i; j--){`
	`if (strcmp(st[j].aaa ,st[j-1].aaa)<0) {`
	`strcpy(t, st[j].aaa);`
	`strcpy(st[j].aaa, st[j-1].aaa);`
	`strcpy(st[j-1].aaa ,t);}}}`

11–15	`double a=0;`
	`for (int i=0; i< N; i++) a+=st[i].c;`
	`a=a/N;`
	`for (int i=0; i< N; i++)`
	`if (st[i].c>a) cout<<st[i].fff<<"\n";`

16–20	`for (int i=0; i< N; i++)`
	`if ((*(st+i)).b%2)`
	`cout<<(*(st+i)).fff<<"\n";`

21–25	`char c='j';char *l;`
	`for (int i=0; i<N; i++){`
	`if (l=strchr(st[i].fff,c))`
	`for (int j=0; j<strlen(l); j++)`
	`st[i].fff[j]-=32;`
	`}`

26–30	`char c='f';char *l;`
	`for (int i=0; i<N; i++){ if (l=strchr(st[i].aaa,c)) {int k=strlen(st[i].aaa);`
	`for (int j=0;j<k/2;j++) {char c=st[i].aaa[j];`
	`st[i].aaa[j]=st[i].aaa[k-j-1];`
	`st[i].aaa[k-j-1]=c;}`
	`}`
	`}`

Завдання 4.2. Створити структуру, специфікація якої наведена нижче (див. табл. 4.2). Визначити функції, що створюють масив структур. Задати критерій відбору.

Таблиця 4.2 – Варіанти завдання 4.2

№	Завдання
1	Структура "Викладач": – прізвище, ім'я, по-батькові; – номер телефону; – кафедра; – дисципліни, що викладає (5 назв). Створити масив структур. Вивести: a) список викладачів вказаної кафедри; b) список викладачів, що викладають задану дисципліну; c) список викладачів в алфавітному порядку.
2	Структура "Поїзд": – номер; – пункт призначення; – час відправлення; – кількість купейних місць, – кількість плацкартних місць, – кількість загальних місць Створити масив структур. Вивести: a) список поїздів, які відходять з заданого пункту відправлення; b) список поїздів, які прямують до заданого пункту призначення та відправляються після вказаної години; c) список поїздів, які відправляються до заданого пункту призначення та мають плацкартні місця.
3	Структура "Абітурієнт": – прізвище, ім'я, по-батькові; – рік народження; – оцінки ЗНО (3); – середній бал атестату. Створити масив структур. Вивести: a) список абітурієнтів, що мають оцінки по ЗНО, нижче вказаного мінімуму; b) список абітурієнтів, у яких середній бал атестату вище заданого значення; c) вибрати вказану кількість n абітурієнтів, що мають найбільшу суму балів.

Продовж. табл. 4.2

№	Завдання
4	Структура "Співробітник": – прізвище, ім'я, по-батькові; – посаду – рік народження; – заробітна плата. Створити масив структур. Вивести: a) список співробітників, заробітна плата нижче вказаного значення; b) список співробітників, що займають певну посаду; c) список співробітників в алфавітному порядку.
5	Структура "Держава": – назва; – столиця; – чисельність населення; – займана площа. Створити масив структур. Вивести: a) список держав, які мають задану чисельність населення; b) список держав, що мають площу більшу за вказану і чисельність населення у вказаному діапазоні; c) список держав, в алфавітному порядку.
6	Структура "Людина": – прізвище, ім'я, по-батькові; – домашня адреса; – номер телефону; – вік. Створити масив структур. Вивести: a) відомості про людину, у яких номер телефону належить певному оператору сотового зв'язку; b) відомості про людину, які живуть на одній вулиці; c) відомості про людину в алфавітному порядку.
7	Структура "Покупець": – прізвище, ім'я, по-батькові; – домашня адреса; – номер телефону; – номер дисконтної картки; – сума витрат.

Продовж. табл. 4.2

№	Завдання
	Створити масив структур. Вивести: a) список покупців в алфавітному порядку; b) список покупців, у яких номер дисконтної картки знаходиться в заданому інтервалі; c) список покупців, чия у яких сума витрат більша за вказане значення.
8	Структура "Людина": – прізвище, ім'я, по-батькові; – рік народження; – зріст; – вага; – стать. Створити масив структур. Вивести: a) відомості про людину, яка має вагу більше заданої і її вік задається певним діапазоном; b) відомості про людину, жіночої статті і певного росту; c) відомості про людину в алфавітному порядку.
9	Структура "Пацієнт": – прізвище, ім'я, по-батькові; – домашня адреса; – номер медичної карти; – діагноз. Створити масив структур. Вивести: a) список пацієнтів, що мають вказаний діагноз; b) список пацієнтів, чий номер медичної картки має останні цифри 34; c) відомості про пацієнтів в алфавітному порядку.
10	Структура "Футбольна команда": – назва; – місто; – кількість гравців; – кількість набраних очок. Створити масив структур. Вивести: a) список команд, що з одного міста; b) список команд, які набрали менше вказаної кількості очок; c) відсортувати команди за назвою.

Продовж. табл. 4.2

№	Завдання
11	Структура "Власник автомобілю": – прізвище, ім'я, по-батькові; – номер автомобілю; – телефон; – колір; – номер техпаспорту. Створити масив структур. Вивести: a) список власників автомобілів заданого кольору; b) список власників автомобілів, які в номері мають певні дві цифри; c) відомості про власників автомобілів в алфавітному порядку.
12	Структура "Студент": – прізвище, ім'я, по-батькові; – домашня адреса; – група; – телефон; – рейтинг. Створити масив структур. Вивести: a) список студентів вказаної групи; b) список N студентів, що мають найвищий рейтинг; c) список навчальної групи в алфавітному порядку.
13	Структура "DVD-диск": – назва фільму; – рік випуску; – жанр; – тривалість; – ціна. Створити масив структур. Вивести: a) відомості про DVD-диск, певного жанру за останні п’ять років; b) відомості про DVD-диск, з найменшою ціною; c) відомості про фільм в алфавітному порядку.
14	Структура "Автомобіль": – марка; – рік випуску; – ціна; – колір.

Продовж. табл. 4.2

№	Завдання
	Створити масив структур. Вивести: a) список автомобілів заданої марки; b) список автомобілів заданої марки, які експлуатуються більше *n* років; c) список автомобілів вказаного року випуску, ціна яких більше вказаної.
15	Структура "Держава": – назва; – кількість державних мов; – державна мова (варіативна частина); – грошова одиниця; – курс валюти відносно $. Створити масив структур. Вивести: a) список держав, які мають більше ніж одну державну мову, мови теж виводити; b) список держав, які мають низький курс відносно $; c) список держав в алфавітному порядку.
16	Структура "Власник автомобіля": – прізвище, ім'я, по-батькові; – номер автомобіля; – номер техпаспорту; – відділення реєстрації ДАІ. Створити масив структур. Вивести: a) список автомобілів, що реєструвалися у певному відділенні ДАІ; b) список автомобілів, що належать певній людині; c) список автомобілів в алфавітному порядку.
17	Структура "Книга": – назва; – автор; – жанр; – рік видання; – кількість сторінок. Створити масив структур. Вивести: a) список книг заданого автора; b) список книг, певного жанру; c) список книг, що видані після заданого року.

Продовж. табл. 4.2

№	Завдання
18	Структура "Фільм": – назва; – режисер; – країна; – принесений прибуток. Створити масив структур. Вивести: a) список фільмів певного режисера; b) список фільмів, які принесли прибуток більше *N*, та які знімалися в Голівуді; c) список фільмів в алфавітному порядку.
19	Структура "Автомобіль": – марка; – серійний номер; – кількість обертів двигуна; – літраж двигуна; – рік випуску. Створити масив структур. Вивести: a) список автомобілів заданої марки; b) список автомобілів, які мають об'єм двигуна більше за 3 л., та які експлуатуються менше *n* років; c) список автомобілів вказаного року випуску, які мають певну цифру в серійному номері.
20	Структура "Басейн": – назва; – адреса; – місто; – дні та час роботи; – телефон адміністратора. Створити масив структур. Вивести: a) список басейнів, що розташовані в певному місті; b) список басейнів, які не працюють пізніше 20.00; c) відсортувати басейни за назвою.

Короткі теоретичні відомості

Структура – це тип, що складається з одного або більше об'єктів, що можливо мають різні типи, об'єднані під одним ім'ям. Важливе використання структур – створення нових типів даних. Тип даних `struct` – один з основних будівельних блоків даних у мові С++. Він надає зручний спосіб об'єднання різних елементів, поєднаних між собою логічним зв'язком. Усі дані, що стосуються одного об'єкта, зібрані разом.

Розглянемо синтаксис оголошення структури на прикладі створення користувацького типу даних (структури) для зберігання дати:

```
struct date
{
     int day; //день
     int month; //місяць
     int year;//рік
     int weekday; //день тижня
     char mon_name[15];// назва місяця
};
```

Створення об'єкта з типом `date` та ініціалізація його при створенні:

```
date my_birthday={20,7,1981,1,"July"};
```

Особливості структур:

1. Опис структури починається зі службового слова `struct`, за яким може прямувати необов'язкове ім'я, що називається ім'ям типу структури. Це ім'я типу структури використовується надалі для створення конкретного об'єкта.

2. За ім'ям типу структури йде укладений у фігурні дужки список елементів структури, з описом типу кожного елемента (елементом структури може бути змінна, масив або структура). Елементи структури відокремлюються один від одного крапкою з комою.

3. За правою фігурною дужкою, що закриває список елементів, може йти список об'єктів. Наприклад, оператор `struct date {...} x, y, z;` визначає змінні `x, y, z` у якості структур описаного типу й приводить до виділення пам'яті.

4. Опис структури, за яким не прямує список об'єктів, не приводить до виділення пам'яті, воно тільки визначає шаблон (форму) структури. Однак, якщо такий опис має ім'я типу (наприклад, `date`), те це ім'я типу може бути використане пізніше при визначенні об'єктів структур.

5. Структуру можна ініціалізувати, помістивши слідом за її визначенням список компонентів, взятих у фігурні дужки.

Структури можуть вкладатися одна в іншу, але самовкладення структур заборонене. Операція доступу до елемента структури `"."` обчислюється зліва направо.

Звертання до елементу структури здійснюється через оператор крапка `(.)` за допомогою конструкції виду: `<ім'я структури>.<ім'я елемента>: myBirthday.day; myBirthday.month;` і т. д.

Також доступ до елемента структури можна здійснювати через вказівник за допомогою операції `"->"` у вигляді: `<вказівник на структуру> "->" <елемент_структури>.`

Звернення `pd->year` і `(*pd).year` еквівалентні. Круглі дужки `(*pd)` необхідні, оскільки операція `"."` доступу до елемента структури старше, ніж `"*"`. Оператор `"."` не може бути перевантажений, на відміну від оператора `"->"`, який, як правило, призначений для роботи із вказівниками.

Дійсне визначення адреси структури за допомогою операції `"&"`, а також присвоювання структури як єдиного цілого. Структури можна передавати як параметр функції і повертати в результаті роботи функції, а також передавати окремі компоненти структури в якості аргументів функції.

Порядок виконання деяких найбільш поширених операцій над елементами структури такого опису:

```
struct {int x; int *y;}
*p; // p – вказівник на структуру
```

1) `(++p)->x` – операція збільшує `p` до доступу до `x`;

2) `(p++)->x` – операція збільшує `p` після доступа до `x` (круглі дужки не обов'язкові, тому що за пріоритетом раніше буде застосована операція `"->"`);

3) `*p->y` – вибирається вміст об'єкту, на який вказує `y`;

4) `*p->y++` – збільшується `y` після обробки того, на що він вказує;

5) `*p++->y` – збільшує `p` після вибрання того, на що вказує `y`;

6) `(*(*p).y)++` – збільшує те, на що вказує `y`.

У мові С++ існує спеціальна унарна операція `sizeof()`, яка повертає розмір свого операнда в байтах. Операндом операції `sizeof()` може бути будь-який вираз: `sizeof(вираз);` Результат операції `sizeof()` має тип `int`.

Розмір структури не дорівнює сумі розмірів її членів. Внаслідок вирівнювання об'єктів різної довжини в структурі можуть з'являтися безіменні "діри". Так, наприклад, якщо змінна типу `char` займає один байт, а `int` – чотири байти, то для структури: `struct Test {char c; int i;};` може знадобитися вісім байт, а не п'ять. Правильне значення повертає операція `sizeof()`.

Приклад виконання роботи

Завдання 4.1. Визначити дію фрагмента програми

```
char c='j';
  for (int i=0; i<N; i++){
       if (st[i].fff[0]==c){
           strcat(st[i].fff,"123");
       }
}
```

Розв'язання

Цей фрагмент програми додає до символьного поля структури число 123, якщо рядок починається з символу 'j':

```
fff: assdss  aaa: ssdhshf  b : 4   c : 4.5
fff: jherhhgb123  aaa: xczvfdg  b : 9   c : 2.4
fff: xcvfd  aaa: hhgeryt  b : 5   c : 4.1
fff: jrhjdfd123  aaa: tyyyyy  b : 7   c : 3.5
```

Завдання 4.2. Скласти програму, яка визначає структуру "Студент": ПІБ, адреса, вік. Вивести на екран дані про певного студента; упорядкувати за алфавітом у порядку зростання.

Розв'язання

1. Постановка задачі

Скласти програму, яка визначає структуру "Студент": ПІБ, адреса, вік. Вивести на екран дані про певного студента; упорядкувати за алфавітом у порядку зростання.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Визначити масив структур st1153.

Дія 2. Ввести значення полів структури.

Дія 3. Ввести ПІБ певного студента.

Дія 4. Для кожного елементу масиву виконати перевірку умови (поле ПІБ масиву співпадає з ПІБ певного студента).

Дія 5. Відсортувати масив структур за полем ПІБ.

Дія 6. Вивести елементи масиву st1153 після сортування.

3. Текст програми

```
#include <iostream>
#include <windows.h>
using namespace std;
struct student{
```

```
    char *fio;
    char *adress;
    int age;};
student enterStruct(){
student st;
cout << "ПІБ: " ;
st.fio=new char [20];
cin.getline(st.fio,20);
cout << "Адреса: " ;
st.adress=new char [20];
cin.getline(st.adress,20);
cout << "Вік " ;
cin>>st.age;
return st;
}
void printStruct(student st1153[],int n){
for (int i=0;i<n;i++){
  cout << "ПІБ: " << st1153[i].fio << endl;
  cout << "Адреса: "    << st1153[i].adress << endl;
  cout << "Вік : " << st1153[i].age    << endl;
}
}
void sortStudent(student st1153[],int n){
student t;
for (int i=0; i< n; i++) {
for (int j= n-1; j>i; j--){
   if (stricmp(st1153[j].fio ,st1153[j-1].fio)<0) {
        t=st1153[j];
        st1153[j]= st1153[j-1];
        st1153[j-1]=t;}}}
}
void printFilter(student st1153[],int n,char * s){
for (int i=0;i<n;i++)
    if (stricmp(st1153[i].fio,s)==0){
      cout << "ПІБ: " << st1153[i].fio << endl;
     cout << "Адреса: "    << st1153[i].adress << endl;
      cout << "Вік : " << st1153[i].age    << endl;
```

```
}
}
int main(){
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
  student st1153[26];
  int n=3;char fio[20];
for (int i=0;i<n;i++){
    st1153[i]=enterStruct();
    cin.ignore();
}
cout<<"\nВведіть ПІБ:";
cin.getline(fio,20);
printFilter (st1153,n,fio);
cout<<"\n Упорядковані дані \n";
sortStudent(st1153,n);
printStruct(st1153,n);
  return 0;
}
```

4. Результати роботи програми

```
ПІБ: Петренко Іван
Адреса: 1 Лінія, д.7
Вік 21
ПІБ: Веселов Максим
Адреса: 7 Поперечна, д.6
Вік 22
ПІБ: Биков Андрій
Адреса: пр. Миру, д.32
Вік 20

Введіть ПІБ:Веселов Максим
ПІБ: Веселов Максим
Адреса: 7 Поперечна, д.6
Вік : 22
```

```
Упорядковані дані
ПІБ: Биков Андрій
Адреса: пр. Миру, д.32
Вік : 20
ПІБ: Веселов Максим
Адреса: 7 Поперечна, д.6
Вік : 22
ПІБ: Петренко Іван
Адреса: 1 Лінія, д.7
Вік : 21
```

Контрольні питання

1. Що таке структура?
2. Який синтаксис оголошення структури?
3. Назвіть особливості структур.
4. Назвіть оператори звернення до певного члена структури та їх відмінності.
5. Назвіть найпоширеніші операції над елементами структури.
6. Що робить унарна операція `sizeof()`?

Робота №5

Розробка та реалізація програм з використанням форматування вводу/виводу, використовуючи консоль та текстовий файл

Мета роботи: оволодіння навичками складання програм з використанням форматування вводу/виводу, використовуючи консоль та текстовий файл, та виконання їх в NetBeans IDE.

Завдання

Завдання 5.1. Визначити дію фрагмента програми за варіантами, наведеними в таблиці 5.1. Визначити вміст файлу `file.txt`, якщо

```
#include <cstdio>
int main()
{  FILE * fl;
    fl=fopen("d:\\file.txt","w");
   double d, x; float f; long lng; int i, y; short s;
```

(Вказівка: `int N` дорівнює номеру варіанта; якщо формат не відповідає типу змінної, поясніть де, і як це впливає на результат).

Таблиця 5.1 – Варіанти завдання 5.1

№	Фрагмент програми

1–5	`s = i = lng = f = d = N*100/3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`d = 3.2; i = 2;x = ( y = d / i ) * N;`
	`fprintf(fl,"x = %f ;y = %d\n", x, y);`

6–10	`d = f = lng = i = s =100/3*N;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`d = 3.2; i = 2;x = ( y = d *N / i )*2;`
	`fprintf(fl,"x = %d ;y = %f\n", x, y);`

Продовж. табл. 5.1

№	Фрагмент програми

11–15	`lng = s = f = i = d =N*100/3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`y = ( x = d / i *N) *2; fprintf(fl,"x = %f ;y = %d\n", x, y);`

16–20	`f = s = d = lng = i = double(100*N)/3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`y = d * ( x = 2.5 / d)*N;fprintf(fl,"x = %f; y = %d\n", x, y);`

21–25	`s = i = lng = f = d = 100/(double)3;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`y = d*N*( x = 2.5 / (d*N)); fprintf(fl,"x = %f; y = %d\n", x, y);`

26–30	`i = s = lng = d = f = (double)(100/3)*N;`
	`fprintf(fl,"s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);`
	`d = 3.2; i = 2;`
	`x = d / N*( y = ( (int)2.9 + 1.1) / d *N);`
	`fprintf(fl,"x = %f y = %d\n", x, y);`

Завдання 5.2. Скласти та виконати програму, яка обробляє інформацію з текстового файлу, згідно завданню, варіанти наведені у таблиці 5.2. Результат записати в файл у формі звіту, обов'язково використовувати форматування тексту.

Вказівка: файл з початковими даними можна створити за допомогою текстового редактора.

Таблиця 5.2 – Варіанти завдання 5.2

№	Варіанти завдання
1–2	Директор школи набирає групу для навчання школярів по факультативній програмі. Навчання платне, загальна вартість курсу *K* грн. Скільки повинен платити кожен учень? Очевидно, ця сума залежить від значення *K* та від кількості учнів. Обчислити і записати в файл таблицю сум, яку повинен внести один учень, якщо група буде складатися з трьох, чотирьох, і т. д. до 20-ти учнів.
3–4	На день народження дитини бабуся відкрила рахунок у банку і поклала на нього 30 доларів. Щороку вона додає 50 доларів. Річний відсоток по банківському рахунку дорівнює 7 %. Яка сума накопичиться до повноліття дитини (до 16-ти років), включаючи останній внесок. Вивести в файл таблицю щорічного стану рахунку.
5–6	Відсоток за банківським вкладом дорівнює 9 %. Якщо покласти в банк суму *N* грн., то ця сума буде щорічно збільшуватися. Як буде змінюватися сума протягом найближчих 10-ти років? Якщо річна інфляція становить 4,5 %, то скільки ж насправді коштуватимуть ці гроші? Вивести в файл таблицю щорічного стану рахунку.
7–8	Нехай у файлі зберігається інформація про багаж пасажира: ПІБ пасажира, кількість речей і загальна вага речей. Написати функції для введення і виведення з файлу загальної інформації про багаж. Знайти пасажирів, що мають не більше двох речей. Знайти кількість пасажирів, кількість речей яких перевершує середню кількість речей. Результати занести в файл.
9–10	Оплата праці няні здійснюється по годинах. За термін до 6-ти годин вона отримує по 25 грн. за годину. Починаючи з 6-ї години роботи, вартість кожної наступної години збільшується на 20 %, тобто 30 грн, потім 36 грн. і т. д. Батьки, вирушаючи на вечірку, хочуть знати суму, яку вони заплатять няньці, але не знають, наскільки затримаються. Обчислити і вивести в файл таблицю оплат послуг няні, починаючи з однієї години до 24-ї години.
11–12	Одружившись, молоде подружжя вирішило відкладати гроші на покупку автомобіля. Чоловік може вкласти щомісяця *M* $, дружина *V* $. Якщо покласти гроші в банк, то за строковим вкладом річний відсоток дорівнює 8 %. Автомобіль мрії має вартість *N* $. Через який термін молоді поїдуть у власному авто? Виведіть в файл таблицю щомісячних накопичень з урахуванням відсотка за банківським вкладом.

Продовж. табл. 5.2

№	Варіанти завдання
13–14	Після закінчення сесії завжди є деяка кількість «боржників». Деканат вирішив провести курси для відстаючих в обсязі 20 годин, і встановив вартість оплати години, що дорівнює 100 грн. Із суми, сплаченої студентами, викладачеві належить 20 %. Знайти, скільки грошей отримає викладач, якщо буде займатися з одним, двома, трьома, і т.д. до *M* студентів. Чи може він отримати за цю роботу суму 5000 грн. Скількох нероб йому доведеться вчити? Для переконливості виведіть таблицю в файл.
15–16	Незнайка вчить англійську мову. У перший день він вивчив два слова, а кожний наступний день зібрався вивчати на два слова більше, ніж в попередній. Знайдіть і виведіть в файл у вигляді таблиці, на який день Незнайко вивчить 100 слів, 200, 300, 400 і т. д. до 1000? В англійській мові близько 50 тис. слів, а термін життя Незнайок приблизно 30 років. Чи встигне до своєї кончини Незнайка вивчити англійську мову? Якщо ні, то скільки життів Незнайки знадобиться, щоб вивчити англійську мову?
17–18	Бабуся вирішила купити телевізор, коли внук подарував їй 500 грн. Вона поклала їх в банк під 19 % річних. Щомісяця на цей же рахунок бабуся вносить 50 грн. Найдешевший телевізор стоїть *K* грн. Через скільки місяців бабуся купить телевізор? Обчислити і вивести в файл стан рахунку помісячно.
19–20	Мається у файлі таблиця "Успішність студентів за рік". Один рядок таблиці відповідає одному студенту і містить такі відомості: ПІБ студента, номер групи, екзаменаційні оцінки за перший семестр, екзаменаційні оцінки за другий семестр. Впорядкувати рядки таблиці за номерами груп і знайти студентів, у яких середній бал за другий семестр нижче, ніж за перший. Результати записати в файл.
21–22	Равлик повзе вгору по дереву з початковою швидкістю *V* м/сек. При цьому він втомлюється, і його швидкість руху падає за прямолінійним законом на 0,1**V* за кожну годину. Обчислити і вивести таблицю в файл, на яку висоту равлик піднімається протягом кожної години. Дізнатися, через скільки годин він зупиниться.

Продовж. табл. 5.2

№	Варіанти завдання
23–24	Фабрика з виробництва капців щорічно збільшує обсяг продажів на 5 відсотків. Собівартість продукції при цьому зменшується на 1 відсоток. У поточному році обсяг продажів склав *N* тис. грн., а собівартість пари капців *C* грн. Обчислити і вивести в файл таблицю збільшення обсягу продажів і зниження собівартості на найближчі *K* років.
25–26	Учень майстра починає з виготовлення в день однієї табуретки. Вдосконалюючись, він кожен день виготовляє на одну табуретку більше, ніж в попередній день. Більше, ніж 20 табуреток в день виготовити не можна. Знайти, на який день учень досягне вершин майстерності, і скільки табуреток він змайструє за весь час навчання. Для переконливості вивести в файл таблицю щоденної продуктивності.
27–28	Фабрика виробляє туфлі, капці і боти. Дані про обсяги збуту продукції кожного виду за минулий рік помісячно зберігаються в файлі. При підведенні підсумків року дирекція вимагає знайти суми накопичувальним підсумком для кожного виду продукції, і зберегти цю інформацію в файлі, а також з'ясувати, в якому місяці має місце найбільший збут кожного виду продукції. Використовувати функції для накопичення підсумків і для пошуку максимуму.
29–30	У файлі зберігається інформація про кількість студентів в тій чи іншій групі кожного курсу інституту з першого по п'ятий (в першому стовпці – інформація про групи першого курсу, у другому – другого і т. д.). На кожному курсі є 8 груп. Визначити середнє число студентів на кожному курсі, визначити на якому курсі найменше студентів.

Короткі теоретичні відомості

Файл – це іменований блок інформації, розташований на носії інформації.

В C++ відсутні оператори для роботи з файлами. Усі необхідні дії виконуються за допомогою функцій, що знаходяться у стандартних бібліотеках. Вони дозволяють працювати з різними пристроями, при цьому файлова система перетворить їх у єдиний абстрактний логічний пристрій – потік.

Існує кілька різновидів файлів. У Windows таких різновидів два: текстовий файл і бінарний (двійковий) файл. Дана класифікація як критерій використовує спосіб зберігання інформації.

Файл можна відкрити в текстовому або у двійковому режимі. Різниця між режимами відкриття полягає в тому, що при відкритті файлу у двійковому режимі інформація буде в тому ж вигляді, у якому вона зберігається на диску або в пам'яті. При текстовому режимі інформація перетворюється до символів та має роздільники, наприклад, табуляція `(\t)` та закінчення рядка `(\n)`.

Організація роботи з файлами засобами мови С. Структура `FILE`

Структура `FILE` – це сутність мови С. Вона визначена у файлі стандартної бібліотеки `<stdio.h>`. Розмір цієї структури та її поля залежать від ОС і від версії компілятора, тому звичайно користуються вказівником на цю структуру `FILE*`, наприклад, `FILE *f;`.

Файловий вказівник – спеціальна змінна, яка автоматично присвоюється відкритому файлу й зберігає поточну позицію у файлі. Вона переміщається по файлу в момент процесів запису й читання.

Дескриптор файлу – унікальний номер, який операційна система привласнює будь-якому відкритому файлу, щоб відрізняти його від інших. Коли файл закривається, система "відбирає" у нього дескриптор. Саме це унікальне число використовується для роботи з конкретним файлом у програмах. Довідатися про дескриптор конкретного файлу можна за допомогою відповідної функції.

Перед початком роботи з файлом його необхідно відкрити за допомогою функції `fopen()` (див. Додаток А). Типи

доступу для цієї функції наведені в додатку Б. Якщо функція відпрацювала успішно, вона повертає вказівник на відкритий файл, а якщо ні, то – нуль. Вказівник на відкритий файл прийнято зберігати в типі даних `FILE*`.

Неформатоване файлове введення/виведення здійснюють за допомогою функцій `fwrite()` (запис у файл) та `fread()` (читання з файлу).

Для форматованого файлового введення/виведення можна використовувати наступні функції:

- `fgetc()` і `fputc()` дозволяють відповідно здійснити введення-виведення символу;
- `fgets()` і `fputs()` дозволяють відповідно здійснити введення-виведення рядка символів;
- `fscanf()` і `fprintf()` дозволяють відповідно здійснити форматоване введення/виведення, вони аналогічні функціям, наведеним у додатку В, тільки виконують дії стосовно файлу.

Позиціонування по файлу здійснюється за допомогою функцій `fseek()` і `ftell()`.

Після закінчення роботи з файлом його треба закрити за допомогою функції `fclose()`. Відсутність цієї команди може призвести до втрати даних у файлі, який був відкритий для запису (дозапису). Якщо не закривати файли, які відкриті для читання, то це може призвести до обмеження доступу до файлу для інших програм. Крім того, у будь-який ОС є обмеження на кількість одночасно відкритих файлів.

Для форматного виводу в різні потоки значень різних типів, відформатованих згідно із заданим шаблоном, використовується функція `printf()` для виводу в консоль та `fprintf()` для виводу у файл.

Використання цієї функції в програмах на C++ у деяких випадках буває більш зручним, ніж застосування стандартних

потоків cout та cin. Прототип цієї функції знаходиться в заголовному файлі stdio.h.

Формат, що вказується при звертанні до функції printf(), виглядає в такий спосіб:

```
printf (Керуючий_рядок,Аргумент1,Аргумент2,...);
```

Керуючий_рядок – рядок символів, що задає формат виводу даних. Крім звичайного тексту, він містить команди перетворення, які починаються із символу "%", за яким ідуть символи й цифри, що задають правила виводу аргументів. Формат визначається в такий спосіб:

```
%[прапорці][ширина][.точність][F|n|h|l|l] перетворення
```

Формат (команда перетворення) починається з обов'язкового знака відсотка. Щоб вставити у виведений текст сам знак відсотка, його потрібно вказати двічі: %%.

Конструкції у квадратних дужках можуть бути відсутні, а символ " | " (вертикальна риса) свідчить про те, що тут може використовуватися одна з перелічених можливостей.

Аргумент1, Аргумент2,... – параметри, що друкуються, які можуть бути змінними, константами або виразами, що обчислюються перед виводом на друк.

Функції scanf() та fscanf() призначені для введення даних. Так само, як і для функцій printf()/fprintf(), для функцій scanf()/fscanf() вказується керуючий рядок і наступний за нею список аргументів. Звертання до цієї функції має вигляд:

```
scanf (Керуючий_рядок,&Ім'я1,&Ім'я2,...,&Ім'яn);
```

Керуючий_рядок – це рядок символів, який задає кількість і типи змінних, що вводяться. Робиться це так: у форматі вказується символ %, за яким іде буква, яка визначає тип, щодо вводимої змінної.

Комбінація %буква є специфікацією перетворення. При введенні використовуються наступні специфікації:

%d – ввести ціле число;

%c – ввести один символ;

%s – ввести рядок символів.

Специфікації перетворення повинні відповідати кількості й типу змінних, що вводяться.

Ім'я1,Ім'я2,...,&Ім'яn – це імена змінних, значення яких треба ввести.

Правила застосування функцій scanf()/fscanf():

1. Якщо потрібно ввести деяке значення й присвоїти його змінній одного з основних типів, то перед іменем змінної потрібно писати символ &, т.б. передавати змінну у функцію по посиланню.

2. Якщо потрібно ввести значення рядкової змінної, то використовувати символ & не потрібно.

При звертанні до функцій scanf()/fscanf() виконання програми припиняється до введення значень зазначених змінних, після чого робота програми продовжується.

Керуючий рядок містить специфікації перетворення, які використовуються для безпосередньої інтерпретації вхідної послідовності. Керуючий рядок може містити:

– пробіли ' ', символи табуляції '\t' і переведення рядка '\n', які просто ігноруються;

– звичайні символи (крім символу "%"), які припускаються співпадаючими із черговими (відмінними від символів, перелічених у попередньому пункті) символами вхідного потоку;

– специфікації перетворення, що складаються із %, необов'язкового символу "пригнічення" присвоювання "*", необов'язкового числа, що задає максимальну ширину поля й символу перетворення.

Специфікація перетворення управляє перетворенням чергового вхідного поля. Результат звичайно міститься в змінну, на яку посилається відповідний параметр. Однак, якщо є прийñóòí ³ì ñèì âî ë "*", тоді вхідне поле пропускається, і ніякого присвоювання не виконується.

Основні символи перетворення:

`d` – десяткове число;

`x` – шістнадцятирічне число;

`o` – восьмеричне число;

`c` – символ;

`s` – рядок символів.

Приклад виконання роботи

Завдання 5.1. Визначити дію фрагмента програми

```
i = s = lng =  f = (double)(1000000/3)*N;
fprintf(fl,"s = %hd i = %d lng = %ld f = %f\n", s, i, lng, f);
d = 3.2;x = d / N*( y = ( 1 + (int)1.1) / d );
fprintf(fl,"x = %f y = %d\n", x, y);
```

Розв'язання

Цей фрагмент програми записує в текстовий файл `file.txt`, наступні дані

```
s = -15712 i = -15712 lng = 10666656 f = 10666656.000000
x = 0.000000 y = 0
```

Оскільки змінна `s` має тип `short` (діапазон -32768…32767), виникає переповнення розрядної сітки, і як наслідок недостовірний результат.

Завдання 5.2. Скласти та виконати програму, яка обробляє інформацію. В текстовому файлі зберігається інформація про зарплату 18 осіб за кожен місяць року (у першому стовпчику – зарплата за січень, у другому – за лютий і т. д.). Скласти програму для розрахунку середньої зарплати за будь-який

місяць. Результат записати в файл у формі звіту, обов'язково використовувати форматування тексту.

Розв'язання

1. Постановка задачі.

Скласти програму для розрахунку середньої зарплати за будь-який місяць.

2. Алгоритм розв'язання задачі.

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Напишемо програму, що створює та заповнює текстовий файл `file.txt`.

Дія 2. Напишемо ще одну програму, яка виконує наступні дії:

Дія 2.1.Відкрити файл `file.txt` для читання та зчитати з нього інформацію, закрити файл;

Дія 2.2.Створимо функцію, що знаходить середнє арифметичне заданого стовпчика;

Дія 2.3.Створимо новий файл `fileNew.txt`, та запишемо в нього назви місяців, зарплату за кожен місяць та середню зарплату за заданий місяць.

3. Текст програм

```
//программа, що створює текстовий файл
#include <cstdio>
#include <cstdlib>
#include <windows.h>
int main()
{FILE *f;double k;    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
char month[12][10]={"січень","лютий","березень","квітень",
"травень","червень","липень","серпень","вересень","жовтень",
"листопад","грудень"};
 f=fopen("file.txt","w");
  for (int j=0;j<12;j++)
```

```
        fprintf(f,"%10s",month[j]);
  fprintf(f,"\n");
  for(int i=0;i<18;i++){
  for (int j=0;j<12;j++){
        k=rand()%2000+2000+1/double(rand()%10+1);
    fprintf(f,"%10.2f",k);}
    fprintf(f,"\n");
  }fclose(f);
    return 0;
}
//программа, що обробляє текстовий файл
#include <cstdio>
#include <cstdlib>
#include <windows.h>
void readFile(float salary[][12],int &n,char s[][10]){
FILE *f=fopen("file.txt","r");
if(f==0){ printf("error read from file");
exit(1); }
for (int j=0;j<12;j++){
fscanf(f,"%s",&s[j]);
}
int i=0,j=0;
while (fscanf(f,"%f",&salary[i][j++])!=EOF){
    if (j==12) {i++;j=0;}
}
n=i; fclose(f);
}
double average(float salary[][12],int n,int m){
    double s=0;
  for (int i=0;i<n;i++)
        s+=salary[i][m];
  return s/n;
}
void writeFile(float salary[][12],int n,char month[][10]){
FILE *f2=fopen("fileNew.txt","w");
  for (int j=0;j<12;j++)
```

```
     fprintf(f2,"%10s",month[j]);
  fprintf(f2,"\n");
  for(int i=0;i<n;i++){
  for (int j=0;j<12;j++)
       fprintf(f2,"%10.2f",salary[i][j]);
       fprintf(f2,"\n"); }
printf("введіть місяць (число): ");int m;
scanf("%d",&m);
fprintf(f2,"\nСередня зарплата за %s місяць \n",month[m-1]);
for (int j=0;j<12;j++)
if (j==m-1)
     fprintf(f2,"%10.2f",average(salary,n,m-1));
else fprintf(f2,"          ");
   fclose(f2);
}
int main()
{char s[12][10];
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    float salary[30][12];int n;
readFile(salary,n,s);
writeFile(salary,n,s);
    return 0;
}
```

4. Результати роботи програми

Через економію печатних сторінок результати роботи програми в методичних вказівках не наводяться. При оформленні роботи результати роботи програми наводяться обов`язково.

Контрольні питання

1. Що таке файл, дескриптор файлу, файловий вказівник?
2. Які різновиди файлів ви знаєте?
3. У яких режимах можна відкрити файл?
4. Що таке структура `FILE` і де вона описана?

5. Як працюють функції `fopen()` та `fclose()`?

6. За допомогою яких функцій можливо здійснити неформатоване файлове введення/виведення при використанні структури `FILE`?

7. За допомогою яких функцій можливо здійснити форматоване файлове введення/виведення при використанні структури `FILE`?

Робота №6

Розробка та реалізація програм для роботи з бінарними файлами

Мета роботи: оволодіння навичками складання програм для роботи з бінарними файлами та виконання їх в NetBeans IDE.

Завдання

Завдання 6.1. Запишіть рядки, які будуть виведені на екран дісплею внаслідок виконання фрагментів, поданих у варіантах табл. 6.1, при наступному початку програми (Вказівка: замість N підставити номер варіанта за списком групи)

```
#include <iostream>
#include<fstream>
#include <cstring>
using namespace std;
int main(){
struct rB{
char name[10];
char ph[5];}; int i,j,nr;
ofstream f; char s[10];
struct {
  bool b;
  union {
       short c;
       short d;};
}a;
f.open("LR9_1.txt",ios::binary);
rB rBk;
strcpy(rBk.name,"AA");  strcpy(rBk.ph,"2222");
f.write((char*)&rBk, sizeof rBk);
strcpy(rBk.name,"BBB");  strcpy(rBk.ph,"333");
```

```
f.write((char*)&rBk, sizeof rBk);
   strcpy(rBk.name,"CC");  strcpy(rBk.ph,"444");
f.write((char*)&rBk, sizeof rBk);
   f.close();
   ifstream ff ("LR9_1.txt",ios::binary);
   nr=sizeof(rB);
   a.c=N; a.d=10*N;
   cout<<a.c<<"  "<<a.d<<"\n";
```

Таблиця 6.1 – Варіанти завдання 6.1

№	Фрагмент програми

1–5	`for(int i=1; i<3;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph;`
	`}`
	`ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.ph<<"\n";`
	`  cout<<"n="<<nr;`
	`ff.close();`
	`  cout<<"  "<<rBk.name<<"\n";`

6–10	`for(int i=0; i<3;i++){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph;`
	`}`
	`cout<<"n="<<nr;`
	`cout<<"  "<<rBk.name<<"\n";`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.ph<<"\n";`
	`ff.close();`

11–15	`for(int i=2; i>1;i--){`
	`  ff.read((char*)&rBk, sizeof rBk);`
	`  cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.ph;`
	`cout<<"n="<<nr;ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

Продовж. табл. 6.1

№	Фрагмент програми

16–20	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.ph<<" n="<<nr;`
	`for(int i=0; i<2;i++){`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

21–25	`for(int i=0; i<2;i++){`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`cout<<" n="<<nr;ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

26–30	`ff.read((char*)&rBk, sizeof rBk);`
	`for(int i=1; i<3;i++){`
	`ff.read((char*)&rBk, sizeof rBk);`
	`cout<<rBk.name<<"  "<<rBk.ph<<"\n";`
	`}`
	`cout<<rBk.ph;`
	`cout<<" n="<<nr;ff.close();`
	`cout<<"  "<<rBk.name<<"\n";`

Завдання 6.2. Скласти та виконати програму, яка обробляє інформацію з файлу. Файл треба створити на основі завдання 4.2. Виконати запити, описані в завданні 4.2.

Вказівка: реалізувати меню з опціями (створити файл, додати один запис у файл, додати декілька записів у файл, видалити файл, видалити один запис у файлі, видалити декілька записів у файлі, вивести вміст файлу на екран, вивести вміст файлу на екран згідно заданих критеріїв).

Короткі теоретичні відомості

Організація роботи з файлами засобами C++. Файлове введення-виведення з використанням потоків.

У мові C++ об'єкти для роботи з файлами називаються потоками (streams). У цьому випадку слово "потік" означає те ж саме, що й "файл" у мові C. Класами для роботи з файлами в мові C++ є `fstream`, `ifstream` та `ofstream`.

Для роботи з файловими потоками необхідно підключити бібліотеку `<fstream.h>` для роботи із двонаправленими потоками, яка визначає класи `ifstream` для роботи із вхідними потоками й `ofstream` для роботи з вихідними потоками.

У бібліотечних файлах потоки описані як класи, тобто являють собою користувацькі типи даних (аналогічно структурам даних). В опис класу, крім даних, додаються описи функцій, що обробляють ці дані (відповідно компонентні дані й компонентні функції (методи)). Звертатися до компонентних функцій (методів) можна також, як і до компонентних даних, за допомогою оператора `.` (крапка) або за допомогою операції `->`. Основні функції (методи) для роботи з файлами наведені в Додатку Г.

Використання файлів у програмі припускає наступні операції:

– створення потоку;

– відкриття файлу й зв'язування його з потоком;

– обмін інформацією (введення/виведення);

– закриття файлу.

Для зв'язку файлу з потоком введення/виведення необхідно об'явити об'єкт класу `fstream`, `ifstream` або `ofstream`, передаючи конструкторові цього об'єкта ім'я необхідного файлу. Далі слід відкрити файл. Наприклад:

```
//створити файловий потік f
ifstream f;
//відкрити файл, який треба зв'язати з потоком
f.open("..\\f.dat",ios::in);
```

Також можна використовувати конструктор з параметром:

```
//створити й відкрити для читання файловий потік f
ifstream f ("..\\f.dat",ios::in);
```

Другим параметром у конструкторові є режим відкриття файлу, перелік яких наведений у Додатку Д.

Після того як файл відкритий, можна зчитувати/записувати дані у файл, використовуючи оператори >> (читання з потоку) і << (виведення у потік), аналогічно стандартним потокам `cin` та `cout`.

Введення/виведення символів у файл/з файлу можливо виконати, використовуючи функції `get()` та `put()`. Зчитати з файлу рядок можливо за допомогою функції `getline()`. Якщо необхідно вводити/виводити такі дані, як структури або масиви, можна використовувати функції `read`() та `write`().

При читанні даних із вхідного файлу необхідно контролювати, чи був досягнутий кінець файлу. Це можна зробити за допомогою методу `eof()`. Якщо в процесі роботи виникне помилкова ситуація, то потоковий об'єкт приймає значення рівне `0`. Для перевірки помилок файлових операцій можна використовувати функцію `fail()`.

Після закінчення роботи з файлом його слід закрити за допомогою методу `close()`. По завершенню програми в деструкторі класу цей метод викличеться автоматично, однак якщо програмі файл більше не потрібний, краще його закрити вручну, оскільки при цьому всі дані, які програма записала у файл, скидаються на диск, і оновлюється запис каталогу для цього файлу.

При роботі з файлами не слід змішувати різні техніки. Наприклад, не потрібно в одній і тій же програмі використовувати функції зі стандартної бібліотеки С і класи – потоки з мови С++.

Приклад виконання роботи

Завдання 6.1. Пояснити, що виконує наступний фрагмента програми, де N=32.

```
ff.read((char*)&rBk, sizeof rBk);
cout<<rBk.name<<"  "<<rBk.ph<<"\n";
ff.read((char*)&rBk, sizeof rBk);

 cout<<rBk.name;
 ff.close();
 cout<<"  "<<rBk.ph<<"\n";
```

Розв'язання

Цей фрагмент програми зчитує інформацію з бінарного файлу та виводить на дисплей.

```
320  320
AA  2222
BBB  333
```

Завдання 6.2. Скласти та виконати програму, яка зберігає дані про студентів (прізвище, ім'я, по батькові, номер групи) у файл, зчитує дані з файлу та виводить на екран, упорядковує дані в алфавітному порядку.

Розв'язання

1. Постановка задачі.

Скласти програму, яка зберігає дані про студентів (прізвище, ім'я, по батькові, номер групи) у файл, зчитує дані з файлу та виводить на екран, упорядковує дані в алфавітному порядку.

2. Алгоритм розв'язання задачі

Дія 1.Виконувати дію 2, дію 3, дію 4, дію 5 в нескінченому циклі.

Дія 2. Якщо обрана дія 2, тоді

Дія 2.1. Визначити масив структур `sd`.

Дія 2.2. Ввести значення полів структури.

Дія 2.3. Ввести ім'я файлу.

Дія 2.4. Додати дані до існуючого файлу або створити новий файл.

Дія 3. Якщо обрана дія 3, тоді:

Дія 3.1. Ввести ім'я файлу.

Дія 3.2. Якщо файл відкривається перейти до дії 3.3, інакше до головного меню.

Дія 3.3. Визначити розмір масиву структур `sd`.

Дія 3.4. Зчитати дані з файлу у масив структур `sd`.

Дія 3.5. Ввести значення полів масиву структур `sd`.

Дія 4. Якщо обрана дія 4, тоді

Дія 4.1. Ввести ім'я файлу.

Дія 4.2. Якщо файл відкривається перейти до дії 4.3, інакше до головного меню.

Дія 4.3. Визначити розмір масиву структур `sd`.

Дія 4.4. Зчитати дані з файлу у масив структур `sd`.

Дія 4.5. Відсортувати масив структур за прізвищем.

Дія 4.6. Вивести елементи масиву після сортування.

Дія 4.7. Запитати у користувача необхідність збереження упорядкованих даних, якщо так, тоді перейти до дії 4.8, інакше до головного меню.

Дія 4.8. Переписати файл

Дія 5. Якщо обрана дія 5, тоді кінець роботи програми.

3. Текст програми

```
#include <iostream>
#include <cstring>
```

```
#include <fstream>
#include <windows.h>
using namespace std;
struct stud{
    char surname[30];
    char name[30];
    char name2[30];
    char grupa[10];};
void enterStruct(stud  sd[], int kol){
    for(int i=0; i < kol; i++){
        cout << "\nПрізвище: ";cin >> sd[i].surname;
         cout << "Ім'я: "; cin >> sd[i].name;
        cout << "По батькові: "; cin >> sd[i].name2;
         cout << "Група: "; cin >> sd[i].grupa; }
}
void fileWrite(char* txt,stud  sd[], int kol) {
 ofstream t(txt,ios::binary | ios::app);
 for(int i=0;i<kol;i++){
        t.write((char*)&sd[i], sizeof sd[i]); }
    t.close(); }
 void printStruct(stud* sd,int kol){
    for(int i = 0; i < kol; i++){
       cout << "\nПрізвище: "<< sd[i].surname;
       cout << "\nІм'я:  "<< sd[i].name;
        cout << "\nПо батькові: "<<sd[i].name2;
        cout << "\nГрупа:  "<< sd[i].grupa<<"\n";}}
int fileOfSize (char *txt) {
    ifstream t(txt,ios::binary);
    if (!t.is_open()){
   cout << "Файл не може бути відкритий  !\n";return 0; }
    t.seekg(0,ios_base::end);
    int k = t.tellg()/sizeof (stud);
    t.close();
    return k;}
void readFile(char *txt,stud* sd,int kol) {
    ifstream t(txt,ios::binary );
    for(int i=0;i<kol;i++){
```

```
  t.read((char*)&sd[i], sizeof sd[i]); }
    t.close();}
void rewriteFile(char* txt,stud* sd,int kol) {
    ofstream t(txt,ios::binary);
    for(int i=0;i<kol;i++){
        t.write((char*)&sd[i], sizeof sd[i]);}
    t.close();}
void sortStud(stud* sd,int kol){
    stud s;
    for(int i=1;i<kol;i++)
    for(int j=kol-1;j>=i;j--)
   if(stricmp(sd[j].surname, sd[j-1].surname) < 0){
    s = sd[j];
    sd[j]=sd[j-1];
    sd[j-1]=s;
    }
   else if(stricmp(sd[j].surname, sd[j-1].surname) ==0){
        if(stricmp(sd[j].name, sd[j-1].name) < 0){
        s = sd[j];
        sd[j]=sd[j-1];
        sd[j-1]=s;
        }
}}
int main(){
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    char fName[80];
    int k = 0;stud  * sd;
   while(1){
        cout<<"0: Exit\n";
        cout<<"1: Додати записи у файл\n";
        cout<<"2: Вивести вміст файлу на екран\n";
        cout<<"3: Сортування за прізвищем\n ";
        cin >> k;
        switch(k){
        case 1:
          cout<<"\nВведіть кількість студентів:  ";
```

```
int zn;
            cin>>zn; sd = new stud[zn];
            enterStruct(sd, zn);
            cin.ignore();
            cout << "\nВведіть ім'я файлу: ";
            cin.getline(fName,80);
            fileWrite(fName,sd,zn);
            break;
        case 2:
          cout << "\nВведіть ім'я файлу: "; cin.ignore();
             cin.getline(fName,80);
            zn=fileOfSize(fName);
            if (zn!=0){sd = new stud[zn];
            readFile(fName,sd,zn);
            cout << "\n Вміст файлу: \n";
             printStruct(sd,zn); cin.ignore();}
           break;
        case 3:
           cout << "\n Сортування даних за прізвищем: \n";
          cout << "\nВведіть ім'я файлу: "; cin.ignore();
             cin.getline(fName,80);
            zn=fileOfSize(fName);
            if (zn!=0){sd = new stud[zn];
            readFile(fName,sd,zn);
            sortStud(sd,zn);
            printStruct(sd,zn);
            cout << "\nЗберегти упорядковані дані? (Y/N): ";
            char tak;cin>>tak;
            if (tak=='Y' || tak =='y')
           rewriteFile(fName,sd,zn);}
            break;
         case 0:
            exit(0);
            break;
        }}
    return 0;
}
```

4. Результати роботи програми.

Через економію друкованих сторінок результати роботи програми в методичних вказівках не наводяться. При оформленні роботи результати роботи програми наводяться обов'язково.

Контрольні питання

1. Які класи використовуються для роботи з файлами?
2. Як можна звертатися до компонентних даних і компонентних функцій?
3. Які операції припускає використання файлів у програмі?
4. Як можна зв'язати файл із потоком введення/виведення?
5. За допомогою яких функцій можна виконати файлові операції читання/запису при використанні потоків?
6. Які функції існують для перевірки коректності файлових операцій при використанні потоків?

Робота №7

Розробка та реалізація програм з використанням директив препроцесора в додатках

Мета роботи: оволодіння навичками складання програм з використанням директив препроцесора в додатках та виконання їх в NetBeans IDE.

Завдання

Завдання 7.1. Визначити дію фрагмента програми за варіантами, які наведені в таблиці 7.1.

Таблиця 7.1 – Варіанти завдання 7.1

№	Фрагмент програми

1–5	`#include <iostream>`
	`using namespace std;`
	`# define mkstr(s) #s`
	`int main()`
	`{ cout<<mkstr(I love C);`
	`return 0;`
	`}`

6–10	`#include <iostream>`
	`using namespace std;`
	`# define concat(a,b) a##b`
	`int main()`
	`{ int ij=30;`
	`cout<<concat(i,j);`
	`return 0;`
	`}`

11–15	`#include <iostream>`
	`using namespace std;`
	`#define ArrFlg 1`
	`int main ()`
	`{`
	`#ifdef ArrFlg`
	`int Arr[30];`

Продовж. табл. 7.1

№	Фрагмент програми

	`cout << "Array created!";`
	`#else`
	`cout << "Array is not defined!";`
	`#endif`
	`return 0;`
	`}`

16–20	`#include <iostream>`
	`using namespace std;`
	`#if 1`
	`int Arr[2];`
	`#else`
	`float Arr[2];`
	`#endif`
	`int main (){`
	
	`#if 1`
	`cout<<"int";`
	`#else`
	`cout<<"float";`
	`#endif`
	
	`return 0;`
	`}`

21–25	`#include <iostream>`
	`using namespace std;`
	`#define fl 3`
	`#if fl==1`
	`int Arr[2];`
	`#elif fl==2`
	`float Arr[2];`
	`#elif fl==3`
	`double Arr[2];`
	`#endif`
	`int main (){`
	
	`#if fl==1`
	`cout<<"int";`

Продовж. табл. 7.1

№	Фрагмент програми

	`#elif fl==2`
	`cout<<"float";`
	`#elif fl==3`
	`cout<<"double";`
	`#endif`
	`return 0;`
	`}`

26–30	`#include <iostream>`
	`using namespace std;`
	`int main ()`
	`{`
	`#ifdef ArrFlg`
	`int Arr[10];`
	`cout << "Array created!";`
	`#else`
	`cout << "Array is not defined!";`
	`#endif`
	`return 0;`
	`}`

Завдання 7.2. На основі завдання 6.2 розробити та виконати проект, який складається з декількох файлів: головного файлу, файлів–заголовків та додаткових модулів. У кожному модулі реалізувати окремі функції. Використовувати умовну компіляцію.

Короткі теоретичні відомості

Препроцесор – це програма, яка робить деякі маніпуляції з початковим текстом програми перед тим, як він піддається компіляції. Препроцесори створюють вхідний текст для компіляторів і можуть виконувати наступні функції:

– обробку макровизначень;

– включення файлів;

– "раціональну" передобробку;

– розширення мови.

Оператор (директива) препроцесора – це один рядок вихідного тексту, що починається із символу #, за яким йдуть назва оператора й операнди. Оператори препроцесора можуть з'являтися в будь-якому місці програми, і їх дія поширюється на весь вихідний файл.

Найбільш поширена дія препроцесора – включення у вихідний текст програми вмісту інших файлів. Для включення тексту з файлу використовується команда `#include`, яка має дві форми запису:

```
#include <ім'я_файлу>// ім'я в кутових дужках
#include "ім'я_файлу"// ім'я в лапках
```

Якщо `ім'я_файлу` дається в кутових дужках, то препроцесор шукає файл у стандартних системних каталогах. Якщо `ім'я_файлу` взяте в лапки, то спочатку препроцесор переглядає поточний каталог користувача, а потім звертається до стандартних системних каталогів.

Коли вихідний текст програми обробляється препроцесором, на місце цієї інструкції ставиться вміст відповідного файлу.

На практиці звичайна ситуація, при якій у програмі використовуються функції, тексти яких зберігаються в окремому заголовному файлі `Header File` з розширенням `.h`. Препроцесор додає тексти всіх функцій у програму з файлу `.h` і як єдине ціле передає на компіляцію.

Оператор `#define` часто використовують для визначення символічних констант. Він може з'явитися в будь-якому місці вихідного файлу, а визначення, що дається йому, має силу, починаючи з місця появи й до кінця файлу. Наприкінці визначення символічної константи (наприкінці оператора `#define`) крапка з комою не ставиться. Наприклад:

```
#define min 1
#define max 100
```

У тексті програми замість констант 1 і 100 можна використовувати відповідно `min` та `max`. Однак текст усередині рядків, символьні константи й коментарі не підлягають заміні, тому що рядки й символьні константи є неподільними лексемами мови С++. Так що після макровизначення `#define YES 1` в операторі `cout<<"YES";` не буде зроблено ніякої макропідстановки.

Препроцесор мови С++ дозволяє перевизначати не тільки константи, але й цілком програмні конструкції. Наприклад, можна написати визначення: `#define forever for(;;)` і потім усюди писати нескінченні цикли у вигляді `forever { <тіло циклу> }`.

За допомогою директиви `#define` також можна створювати макроси, які використовуються при впровадженні.

Директиви умовної компіляції, дозволяють генерувати програмний код залежно від виконання певних умов. Умовна компіляція забезпечується в мові С++ набором команд, які, по суті, керують не компіляцією, а препроцесорною обробкою. Загальна структура застосування директив умовної компіляції така:

```
#if/#ifdef/#ifndef <константний_вираз або ідентифікатор>
   <текст_1>
#else //необов'язкова директива
   <текст_2>
#endif
```

`текст_1` включається в текст, який компілюється, тільки при істинності умови, що перевіряється. Якщо умова хибна, тоді при наявності директиви `#else` на компіляцію передається `текст_2`. Якщо директива `#else` відсутня, тоді весь текст від `#if` до `#endif` при хибній умові пропускається.

Директива `#if` перевіряє значення константного цілочисельного виразу. Якщо воно відмінне від нуля, то вважається,

що умова, яка перевіряється істина. У директиві #ifdef перевіряється, чи визначений за допомогою команди #define до теперішнього моменту ідентифікатор, розташований після #ifdef. Якщо ідентифікатор визначений, то текст_1 використовується компілятором. У директиві #ifndef перевіряється зворотна умова – дійсним вважається невизначеність ідентифікатора, тобто той випадок, коли ідентифікатор не був використаний у команді #define або його визначення було скасовано командою #undef.

Директиву #undef зручно використовувати при розробці великих програм, коли вони збираються з окремих "шматків тексту", написаних у різний час або різними програмістами. У цьому випадку можуть зустрітися однакові позначення різних об'єктів. Щоб не змінювати вихідних файлів текст, що включається, можна "обрамляти" відповідними директивами #define - #undef і тим самим усувати можливі помилки.

Для організації мультирозгалужень під час обробки препроцесором вихідного тексту програми введена директива #elif <константний_вираз>, яка є скороченням конструкції #else#if. Препроцесор обробляє завжди тільки одну з ділянок тексту, виділених командами умовної компіляції.

Приклад виконання роботи

Завдання 7.1. Визначити дію фрагмента програми

```
#include <stdio.h>
#define print(x) printf(#x"=%10.5f\n",x);
using namespace std;

int main()
{float a=5.34564, d=235.64456345;
    print(a);print(d);
    return 0;
}
```

Розв'язання

У тілі макросу перед параметрами можна задавати символ #, це означає що при розширенні макросу параметри перетворюються в рядок символів.

Цей фрагмент програми виводить на екран:

```
a=   5.34564
d= 235.64456
```

Завдання 7.2. На основі завдання 6.2 розробити та виконати проект, який складається з декількох файлів: головного файлу, файлів–заголовків та додаткових модулів. У кожному модулі реалізувати окремі функції. Використовувати умовну компіляцію.

Розв'язання

1. Постановка задачі.

Скласти програму, яка визначає структуру "Студент": ПІБ, адреса, вік. Вивести на екран дані про певного студента; упорядкувати за алфавітом у порядку зростання.

2. Алгоритм розв'язання задачі.

Алгоритм розв'язання задачі співпадає з алгоритмом розв'язання задачі 6.2 та через економію друкованих сторінок в методичних вказівках не наводяться. При оформленні роботи алгоритм розв'язання задачі наводиться обов'язково.

3. Текст програм

main.cpp

```
#include <iostream>
#include <windows.h>
#include "student.h"
using namespace std;
int main(){
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
```

```
    char fName[80];
    int k = 0;stud  * sd;
   while(1){
        cout<<"0: Exit\n";
        cout<<"1: Додати записи у файл\n";
        cout<<"2: Вивести вміст файлу на екран\n";
        cout<<"3: Сортування за прізвищем\n ";
        cin >> k;
        switch(k){
        case 1:
           cout<<"\nВведіть кількість студентів:  ";
int zn;
         cin>>zn; sd = new stud[zn];
         enterStruct(sd, zn);
         cout << "\nВведіть ім'я файлу: ";cin.ignore();
         cin.getline(fName,80);
         fileWrite(fName,sd,zn);
         break;
        case 2:
         cout << "\nВведіть ім'я файлу: ";cin.ignore();
         cin.getline(fName,80);
         zn=fileOfSize(fName);
         if (zn!=0){sd = new stud[zn];
         readFile(fName,sd,zn);
         cout << "\n Вміст файлу: \n";
         printStruct(sd,zn);getchar();}
         break;
        case 3:
         cout << "\n Сортування даних за прізвищем: \n";
         cout << "\nВведіть ім'я файлу: ";cin.ignore();
         cin.getline(fName,80);
         zn=fileOfSize(fName);
         if (zn!=0){sd = new stud[zn];
         readFile(fName,sd,zn);
         sortStud(sd,zn);
         printStruct(sd,zn);
         cout << "\nЗберегти упорядковані дані? (Y/N): ";
```

```
            char tak;cin>>tak;
            if (tak=='Y' || tak =='y')
            rewriteFile(fName,sd,zn);}
            break;
         case 0:
            exit(0);
            break;
        }}
    return 0;
}
```

student.h

```
#ifndef STUDENT_H_INCLUDED
#define STUDENT_H_INCLUDED
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;
struct stud{
    char surname[30];
    char name[30];
    char name2[30];
    char grupa[10];};
void enterStruct(stud *, int );
void fileWrite(char* ,stud *, int );
 void printStruct(stud* ,int );
int fileOfSize (char *);
void readFile(char *,stud* ,int );
void rewriteFile(char* ,stud* ,int );
void sortStud(stud* ,int );
#endif // STUDENT_H_INCLUDED
```

student.cpp

```
# include "student.h"
void enterStruct(stud  sd[], int kol){
    for(int i=0; i < kol; i++){
        cout << "\nПрізвище: ";cin >> sd[i].surname;
         cout << "Ім'я: "; cin >> sd[i].name;
```

```
        cout << "По батькові: "; cin >> sd[i].name2;
        cout << "Група: "; cin >> sd[i].grupa; }
}
void fileWrite(char* txt,stud  sd[], int kol) {
 ofstream t(txt,ios::binary | ios::app);
 for(int i=0;i<kol;i++){
        t.write((char*)&sd[i], sizeof sd[i]); }
    t.close(); }
 void printStruct(stud* sd,int kol){
    for(int i = 0; i < kol; i++){
       cout << "\nПрізвище: "<< sd[i].surname;
        cout << "\nІм'я:  "<< sd[i].name;
        cout << "\nПо батькові: "<<sd[i].name2;
        cout << "\nГрупа:  "<< sd[i].grupa<<"\n";}}
int fileOfSize (char *txt) {
    ifstream t(txt,ios::binary);
    if (!t.is_open()){
   cout << "Файл не може бути відкритий  !\n";return 0; }
    t.seekg(0,ios_base::end);
    int k = t.tellg()/sizeof (stud);
    t.close();
    return k;}
void readFile(char *txt,stud* sd,int kol) {
    ifstream t(txt,ios::binary );
    for(int i=0;i<kol;i++){
  t.read((char*)&sd[i], sizeof sd[i]); }
    t.close();}
void rewriteFile(char* txt,stud* sd,int kol) {
    ofstream t(txt,ios::binary);
    for(int i=0;i<kol;i++){
        t.write((char*)&sd[i], sizeof sd[i]);}
    t.close();}
void sortStud(stud* sd,int kol){
    stud s;
    for(int i=1;i<kol;i++)
    for(int j=kol-1;j>=i;j--)
      if(stricmp(sd[j].surname, sd[j-1].surname) < 0){
```

```
    s = sd[j];
     sd[j]=sd[j-1];
     sd[j-1]=s;
    }
   else if(stricmp(sd[j].surname, sd[j-1].surname) ==0){
         if(stricmp(sd[j].name, sd[j-1].name) < 0){
        s = sd[j];
         sd[j]=sd[j-1];
        sd[j-1]=s;
       }
}}
```

4. Результати роботи програми

Через економію друкованих сторінок результати роботи програми в методичних вказівках не наводяться. При оформленні роботи результати роботи програми наводяться обов'язково.

Контрольні питання

1. Що таке препроцесор?
2. Що таке директива препроцесора?
3. Назвіть основні директиви препроцесора.
4. Що робить директива препроцесора `#define`?
5. Для чого використовується директива препроцесора `#elif`?
6. Що робить директива препроцесора `#undef`?

Робота №8

Розробка та реалізація програм для роботи з однонаправленими списками

Мета роботи: оволодіння навичками складання програм для роботи з однонаправленими списками та виконання їх в NetBeans IDE.

Завдання

Завдання 8.1. Запишіть рядки, які будуть виведені на екран внаслідок виконання фрагментів, поданих у варіантах (див. табл. 8.1), при наступному початку програми (Вказівка: замість N підставити номер варіанта за списком групи)

```
#include <stdio.h>
#include <string.h>
 struct lnk{
    char name[10];
    int ph;
    lnk *next;
    };
int main(){
int i, nr; short int n; short int *k, *p;
lnk *cR,*fst,*p1; char *nAr[3];  int pAr[3];
scanf("%hd",&n);k=&n;p=k;*p=*p+2;
printf("%p %hd\n",k,*p);
nAr[0]="AAA";nAr[1]="BBBB";nAr[2]="CCCC";
pAr[0]=2222;pAr[1]=333;pAr[2]=4444;
nr=sizeof (lnk);
fst=NULL;
for (i=0; i<3;i++){
   cR = new lnk;
    strcpy(cR->name,nAr[i]);
    cR->ph=pAr[i];
    cR->next=fst; fst=cR;
}
```

Таблиця 8.1 – Варіанти завдання 8.1

№	Фрагмент програми

1–5	`fst->next=fst->next->next;`
	`printf("%d \n",cR->ph); cR=fst;`
	`while (cR!=NULL){`
	`  printf("%s %d\n",cR->name,cR->ph);`
	`  cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf(" %s\n",fst->name);`

6–10	`fst=fst->next;`
	`printf("%d \n",cR->ph);cR=fst;`
	`while ( cR!=NULL){`
	`printf("%s  %d \n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

11–15	`p1=new lnk;`
	`strcpy(p1->name,nAr[0]);p1->ph=pAr[0];`
	`p1->next=fst->next; fst->next=p1;`
	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=2;i>0;i--){`
	`printf("%s  %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

16–20	`p1=new lnk;`
	`strcpy(p1->name,nAr[1]);p1->ph=pAr[1];`
	`p1->next=fst->next; fst->next=p1;`
	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=0;i<3;i++){`
	`printf("%s  %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf("  %s\n",fst->name);`

Продовж. табл. 8.1

№	Фрагмент програми

21–25	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=1;i<3;i++){`
	`printf("%s %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf(" %s\n",fst->name);`

26–30	`printf("%d\n",cR->ph);cR=fst;`
	`for (i=2;i>=0;i--){`
	`printf("%s %d\n",cR->name,cR->ph);`
	`cR=cR->next;`
	`}`
	`printf("nr=%d",nr);`
	`printf(" %s\n",fst->name);`

Завдання 8.2. На основі завдання 6.2 розробити та виконати програму, яка буде зчитувати інформацію з файлу у динамічну зв'язану структуру – зв'язаний список. Масиви не використовувати!

Короткі теоретичні відомості

Список – це структура (складна змінна, клас), яка містить звичайні змінні (поля даних) і адреси наступної такої структури (вказівник на неї) у випадку односпрямованого списку. У випадку двоспрямованого списку структура містить два вказівники – на наступну й попередню структури. Кожний елемент списку містить інформаційне поле (або поля) і посилальне поле. У першому елементі списку посилальне поле містить вказівник на наступну структуру, а в останньому елементі списку посилальне поле містить нулі (`NULL`). У список легко вставити елемент або видалити елемент зі списку, відкоригувавши значення посилального поля

попереднього (або наступного) елемента. Для роботи зі списком звичайно заводять декілька вказівників: на перший елемент, на поточний елемент, на останній елемент.

Динамічна реалізація односпрямованого списку заснована на динамічному виділенні й звільненні пам'яті для елементів списку. Логічна послідовність елементів списку створюється посилальними полями з адресами наступних елементів (останній елемент має порожнє посилання NULL).

Для зручності реалізації вважається, що список завжди містить хоча б один елемент–заголовок з адресою першого реального елемента списку. Це дозволяє уніфікувати процедури додавання й видалення крайніх елементів і усунути деякі перевірки. Адреса елемента–заголовка задається змінною–вказівником Head. Ця змінна встановлюється при початковому створенні списку й надалі не змінюється. Для реалізації основних дій використовуються допоміжні посилальні змінні.

Необхідні оголошення:

```
struct Listitem {
int Info;
Listitem *Next;
};
Listitem *Head;
```

Створення порожнього списку включає:

– виділення пам'яті під заголовок:

```
Head = new Listitem;
```

– встановлення порожньої посилальної частини заголовка:

```
Head->next = NULL;
```

Пошук заданого елемента включає:

– встановлення допоміжного вказівника на адресу першого елемента списку;

– організацію циклу проходу за списком із завершенням або по співпадінню інформаційної частини елемента із

заданим значенням, або по досягненню кінця списку. Після завершення циклу необхідно перевірити значення допоміжного вказівника й зробити висновок про успішність пошуку.

```
Listitem *Current = Head->Next;
while (Current != NULL && Current->Info != Info)
  Current=Current->Next;
if(Current==NULL)
  {…
  //нічого не знайдене;
  }
else
  {…
//елемент знайдений;
  }
```

Видалення заданого елемента включає:

– пошук елемента, що видаляється, з визначенням адреси елемента–попередника. Для цього вводиться ще одна допоміжна посилальна змінна `Previous`, ініційована адресою заголовка, що й змінює своє значення усередині циклу. Якщо елемент, який видаляється, знайдений, тоді змінюється посилальна частина його попередника:

```
Previous->Next = Current->Next;
```

– елемент, який видаляється, обробляється необхідним чином, тобто або звільняється займана ним пам'ять, або він включається в допоміжний список.

Додавання нового елемента після заданого включає:

– пошук заданого елемента за допомогою допоміжного вказівника `Current`;

– виділення пам'яті для нового елемента за допомогою ще одного вказівника `Tmp`;

– формування полів нового елемента, зокрема – настроювання посилальної частини:

```
Tmp->Next = Current->Next;
```

– зміна посилальної частини поточного елемента на адресу нового елемента:

```
Current->Next = Tmp;
```

Додавання нового елемента перед заданим включає:

– пошук заданого елемента з одночасним визначенням елемента–попередника (використовуються вказівники `Current` і `Previous`);

– виділення пам'яті для нового елемента за допомогою ще одного вказівника `Tmp`;

– формування полів нового елемента, зокрема – настроювання посилальної частини:

```
Tmp->Next = Current;
```

– зміна посилальної частини елемента-попередника на адресу нового елемента:

```
Previous->Next = Tmp;
```

Якщо при використанні списку часто доводиться додавати або видаляти елементи наприкінці списку, то для зменшення витрат на перегляд усього списку можна ввести другий основний вказівник – на останній елемент списку. Це потребує зміни функцій видалення й додавання для відстеження цього вказівника.

Часто використовується впорядкований різновид односпрямованого списку, в якому елементи вишиковуються відповідно до заданого порядку, наприклад: цілі числа по зростанню або текстові рядки за алфавітом. Для таких списків змінюється процедура додавання: новий елемент повинен вставлятися у відповідне місце для збереження порядку елементів. Наприклад, якщо порядок елементів визначається цілими числами по зростанню, то при пошуку підходящого місця треба знайти перший елемент, більший заданого й виконати вставку перед цим елементом.

Приклад виконання роботи

Завдання 8.1. Визначити дію фрагмента програми

```
printf("%s\n",cR->name);p1=fst->next;
while (p1->next){
 printf("%s  %d\n",p1->name,p1->ph);
 p1=p1->next;
}
printf("nr=%d",nr);
printf(" %s  %d\n",p1->name,p1->ph);
```

Розв'язання

Цей фрагмент програми виводить на екран у першому рядку вміст змінної k, а саме адресу, та значення за вказівником p, що вказує на n. У другому рядку виводиться значення інформаційного поля останнього елементу списку, далі виводяться значення двох інформаційних полів попереднього елементу. В останньому рядку наведено розмір елементу списку, та значення полів першого елементу списку.

```
0028FF02 6
CCCC
BBBB  333
nr=20 AAA  2222
```

Завдання 8.2. Розробити програму, яка оброблює лінійний список: видалити всі елементи, що більші за середнє арифметичне значення.

Розв'язання

1. Постановка задачі.

Розробити програму, яка оброблює лінійний список: видалити всі елементи, що більші за середнє арифметичне значення.

2. Алгоритм розв'язання задачі.

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Створити список.

Дія 2. Вивести список на екран.

Дія 3. Знайти середнє арифметичне значення елементів списку.

Дія 4. Повторювати доки не кінець списку:

Дія 4.1.Знайти адресу елементу списку, значення інформаційного поля якого більше за середнє арифметичне значення;

Дія 4.2.Якщо адреса не нуль, тоді видалити елемент списку за цією адресою;

Дія 5. Вивести змінений список на екран.

Дія 6. Видалити весь список.

3. Текст програми

```
#include <iostream>
#include <stdlib.h>
using namespace std;
struct element {
    int x;
    element *next;
};
void add(int x, element *head){
    element * t = new element;
    t->x = x;
    t->next = head->next;
    head->next = t;
}
void print(element *head) {
    cout << "------------------------" << endl;
    element * t = head->next;
    while (t != NULL) {
        cout << t->x << " ";
        t = t->next;
    }
   cout << endl << "------------------------" << endl;
}
```

```
element * findAbove(element *head, float value) {
    element *p;
    p = head;
     while(p && (p->x <= value)) {
          p = p->next;
    }
    return p;
}
float average(element *head) {
     element *p;
     int sum=0;int k=0;
    p = head->next;
while (p) {
    sum+=p->x;k++;
    p=p->next;
}
return float(sum)/k;
}
void deleteList(element *head) {
while (head->next) {
    element * t = head->next;
    head->next = t->next;
    delete t;
}
}
void deleteElement(element *p) {
    p->x = p->next->x;
    element *t = p->next;
    p->next = p->next->next;
    delete t;}
int main() {
element *head = new element;
int k;
for (int i=0;i<10;i++){
    k=rand()%10;
    add(k, head);
    }
print(head);
```

```
float av=average(head);
cout<<"\n average="<<av<< endl;
element *temp=head->next;
while (temp!=NULL){
    temp = findAbove(temp, av);
   if (temp) deleteElement(temp);
   }
print(head);
deleteList(head);
    return 0;
}
```

4. Результати роботи програми

```
------------------------
4 2 8 8 4 9 0 4 7 1
------------------------
average=4.7
------------------------
4 2 4 0 4 1
------------------------
```

Контрольні питання

1. Що таке список і елемент списку?

2. У чому відмінність між односпрямованим і двоспрямованим списком?

3. Як створити порожній список?

4. Як відбувається пошук заданого елемента списку?

5. Як відбувається видалення заданого елемента списку?

6. Як відбувається додавання нового елемента списку після заданого елемента?

7. Як відбувається додавання нового елемента списку перед заданим елементом?

Робота №9

Розробка та реалізація програм з використанням рекурсивних алгоритмів

Мета роботи: оволодіння навичками складання програм з використанням рекурсивних алгоритмів та виконання їх в NetBeans IDE.

Завдання

Завдання 9.1. Визначити дію фрагмента програми за варіантами, які наведені в таблиці 9.1 (Вказівка: замість `N` підставити номер варіанта за списком групи).

Таблиця 9.1 – Варіанти завдання 9.1

№	Фрагмент програми

1–5	`#include <iostream>`
	`using namespace std;`
	`int f(int n) {`
	`  if (n < 10)  return n;`
	`  else  return n % 10 + f(n / 10);`
	`  }`
	`int main() {`
	`  int ch=1235+N;`
	`  cout<<"f="<<f(ch);`
	`return 0;`
	`}`

6–10	`#include <iostream>`
	`using namespace std;`
	`int f(int n, int i) {`
	`  if (n <= 1)  return false;`
	`  else if (n == 2)  return true;`
	`  else if (n % i == 0)  return false;`
	`  else`
	`  if (i < n / 2)  return f(n, i + 1);`
	`  else  return true;`

Продовж. табл. 9.1

№	Фрагмент програми

	}
	int main() {
	cout<<"f="<<f(10+N,2)<<"\n";
	cout<<"f="<<f(11,2)<<"\n";
	cout<<"f="<<f(12,2)<<"\n";
	return 0;
	}

11–15	#include <iostream>
	using namespace std;
	void f(int n, int k) {
	if (k > n / 2) {
	cout<<n<<" ";
	return;
	}
	if (n % k == 0) {
	cout<<k<<" ";
	f(n / k, k);
	}
	else f(n, ++k);
	}
	int main() {
	f(125+N,2);
	return 0;
	}

16–20	#include <iostream>
	using namespace std;
	int f(int n, int k) {
	if (n==0) return k;
	else f(n/10, k*10+n%10);
	}
	int main() {
	cout<<"f="<<f(N,0)<<"\n";
	cout<<"f="<<f(125,0)<<"\n";cout<<"f="<<f(4568,0)<<"\n";
	return 0;
	}

21–25	#include <iostream>
	#include <cmath>
	using namespace std;
	float f(int n) {
	if (n <=1) return 1;
	else return (1/sqrt(n)+f(--n));

Продовж. табл. 9.1

№	Фрагмент програми

	}
	int main() {int n=10-N%10;
	cout<<"f="<<f(n)<<"\n";
	return 0;
	}

26–30	#include <iostream>
	#include <limits.h>
	using namespace std;
	int f() {int a;
	cin >>a;
	if (a ==0) return INT_MIN;
	else { int t=f();
	return (a>t)? a : t;}
	}
	int main() {
	cout<<"f="<<f()<<"\n"; return 0;}

Завдання 9.2. Скласти програму знаходження наступних величин за варіантами, які наведені в таблиці 9.2. Замість циклічних операторів використовувати рекурсію!

Таблиця 9.2 – Варіанти завдання 9.2

№	Варіанти завдання
1–2	Знайти найбільший спільний дільник (НСД) чисел x і y. $НСД(x,y)=\begin{cases} x, якщо\ x=y \\ НСД(x-y,y), якщо\ x>y \\ НСД(x,y-x), якщо\ x<y \end{cases}$
3–4	Використовуючи рекурсивну функцію, знайти n-ий елемент послідовності $x_n = 5* x_{n-1}$, $x_0 = 1$.
5–6	Використовуючи рекурсивну функцію, знайти суму перших n елементів послідовності $x_n = 3* x_{n-1}+1$, $x_0 = 0$.
7–8	Використовуючи рекурсивну функцію, знайти добуток перших n елементів послідовності $x_n = 3* x_{n-1}+1$, $x_0 = 1$.
9–10	Вводиться з клавіатури послідовність ненульових дійсних чисел, яка закінчується `0`. Використовуючи рекурсивну функцію, знайти кількість від'ємних чисел.

Продовж. табл. 9.2

№	Варіанти завдання
11–12	Нехай з клавіатури вводиться деяка послідовність символів (враховуючи пробіл). Використовуючи рекурсивну функцію, знайти кількість латинських букв.
13–14	Нехай з клавіатури вводиться деяка послідовність символів (враховуючи пробіл). Використовуючи рекурсивну функцію, знайти кількість цифр.
15–16	Нехай з клавіатури вводиться деяка послідовність символів (враховуючи пробіл). Використовуючи рекурсивну функцію, знайти суму кодів символів.
17–18	Нехай з клавіатури вводиться деяка послідовність символів. Описати рекурсивную функцію, яка перетворює послідовність цифр, що знаходяться серед символів в ціле число. Наприклад, водиться послідовність `апр456ро 78`, отримаемо число `45678`.
19–20	Вводиться з клавіатури послідовність ненульових дійсних чисел, яка закінчується `0`. Використовуючи рекурсивну функцію, знайти середнє арифметичне чисел.
21–22	Використовуючи рекурсивну функцію, обчислити значення за формулою, де *n* вводиться з клавіатури: $\sqrt{2+\sqrt{4+\cdots+\sqrt{2n}}}$
23–24	Використовуючи рекурсивну функцію, обчислити значення за формулою, де *n* вводиться з клавіатури: $\sqrt{1+\sqrt{3+\cdots+\sqrt{2n+1}}}$
25–26	Використовуючи рекурсивну функцію, обчислити значення за формулою, де *n* вводиться з клавіатури: $\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3+\cdots+\cfrac{1}{n}}}}$

Продовж. табл. 9.2

№	Варіанти завдання
27–28	Використовуючи рекурсивну функцію, обчислити значення за формулою, де n вводиться з клавіатури: $\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{5+\cdots+\cfrac{1}{2n+1}}}}$
29–30	Використовуючи рекурсивну функцію, перевернути рядок (вивести в зворотному порядку).

Короткі теоретичні відомості

Рекурсивною називається функція, якщо під час її виконання відбувається повторний її виклик: безпосередньо в тілі функції (прямий виклик) або шляхом виклику ланцюжка інших функцій, які містять звернення до іскомої функції (непрямий виклик). Всі функції, які створюють ланцюжок, також називаються рекурсивними.

Коли функція рекурсивно викликається, в стеку створюється копія значень її локальних змінних та параметрів і функція виконується з початку. При повторному виклику цей процес повторюється. Глибина рекурсії, тобто кількість викликів, у C++ не обмежується. Реально вона залежить від ресурсів пам'яті (розміру стека).

В загальному випадку рекурсивність – це не властивість самої функції, а властивість її опису. Рекурсивна функція, як правило, коротша і наочніша за нерекурсивну, але при виконанні вимагає більше часу і пам'яті за рахунок повторних звертань до самої себе і дублювання локальних змінних. Необхідно добре розуміти, що кожний черговий рекурсивний виклик приводить до утворення нової копії локальних об'єктів

функції і усі ці копії, що відповідають ланцюжкові активізованих і незавершених рекурсивних викликів, існують незалежно один від одного та зберігаються у стеку. Це може привести до так званого "вибуху" стека. "Вибух" стека означає, що для запису локальних змінних функції не вистачає пам'яті, яка відводиться під стек.

У рекурсивних функціях можна виділити дві серії кроків:

– рекурсивне занурення функції доти, доки вибраний параметр не досягне граничного значення. Якщо таке граничне значення не задати, рекурсивна функція створить безкінечну послідовність викликів самої себе;

– рекурсивний вихід доти, доки вибраний параметр не досягне початкового значення. Це забезпечує отримання проміжних та кінцевого результатів.

Найбільш типові приклади рекурсивних функцій – обчислення факторіала і зведення в ступінь.

```
// обчислення факторіала n
int fact(int n) {
if (n==0) return 1;
else
return n*fact(n-1);
}
```

Зазвичай рекурсивні функції дещо зменшують розмір коду і підвищують ефективність використання пам'яті в порівнянні зі своїми ітеративними аналогами, проте основна перевага рекурсивних функцій полягає в спрощенні і наочності алгоритмів.

Приклад виконання роботи

Завдання 9.1. Визначити дію фрагмента програми

```
#include <iostream>
#include <limits.h>
using namespace std;
```

```
int  f() {int n;
      cin >>n;
      if (n ==0) return INT_MAX;
      else { int t=f();
            return (n<t)? n : t;}
 }
int main() {
cout<<"f="<<f()<<"\n";return 0;}
```

Розв'язання

Цей фрагмент програми знаходить мінімальне значення послідовності чисел. Послідовність буде завершена, якщо введено з клавіатури 0.

```
6
3
5
1
9
0
f=1
```

Завдання 9.2. Використовуючи рекурсивну функцію, обчислити кількість проміжків у рядку, який зчитується з консолі.

Розв'язання

1. Постановка задачі.

Використовуючи рекурсивну функцію, обчислити кількість проміжків у рядку, який зчитується з консолі.

2. Алгоритм розв'язання задачі.

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Зчитати рядок `sentence` з консолі.

Дія 2. Повторювати наступні дії :

Дія 2.1. Якщо поточний символ рядка `sentence` дорівнює `\0`, то кінець рекурсії;

Дія 2.2. Якщо поточний символ рядка sentence дорівнює не пробіл, тоді перейти до наступного символу;

Дія 2.3. Якщо поточний символ рядка `sentence` дорівнює пробіл, тоді збільшити результат на 1 та перейти до наступного символу;

Дія 3. Вивести результат роботи рекурсивної функції на екран.

3. Текст програми

```
#include <cstdio>
int countW(char *str){
     if (*str == '\0' )
          return 0;
  if (*str !=' ') return countW(++str);
            else  return countW(++str)+1;
}
int main(){
char sentence[80];
printf( "Enter text:\n" );
gets( sentence);
printf( "Result:\n" );
printf("%d",countW(sentence));
return 0;
}
```

4. Результати роботи програми

```
Enter text:
word w o r d
Result:
4
```

Контрольні питання

1. Яка функція називається рекурсивною?
2. Що таке прямий та непрямий виклик рекурсії?
3. Що таке глибина рекурсії?
4. Назвіть серії кроків у рекурсивній функції.
5. Назвіть переваги рекурсивного опису функцій.
6. Назвіть особливості описання рекурсивної функції.

Робота №10

Розробка та реалізація програм з використанням бітових операцій

Мета роботи: оволодіння навичками складання програм з використанням бітових операцій та виконання їх в NetBeans IDE.

Завдання

Завдання 10.1. Визначити дію фрагмента програми за варіантами, які наведені в таблиці 10.1 (Вказівка: замість `N` підставити номер варіанта за списком групи).

Таблиця 10.1 – Варіанти завдання 10.1

№	Фрагмент програми

1–5	`#include<stdio.h>`
	`#include <limits.h>`
	`int main ()`
	`{size_t len = sizeof(int) * CHAR_BIT;`
	`unsigned int value=N;`
	`const unsigned int mask = 1 << len-1;`
	`printf ("\nThe bitwise representation is: ");`
	`for (int i = 0; i < len; i++)`
	`{`
	`putchar( mask & value ? '1' : '0');`
	`value <<=1;`
	`if ( (i+1) % 8 == 0)`
	`putchar (' ');`
	`}`
	`return 0;`
	`}`

6–10	`#include<stdio.h>`
	`int main () {`
	`unsigned int a= N & 0xFFFE;`
	`int b;`
	`printf ("The shifted number is %u\n", a);`

Продовж. табл. 10.1

№	Фрагмент програми

	printf ("Enter the shift direction: left (0) and right (1) \n");
	scanf ("%d", &b);
	if (!b)
	a = a << ((N^3)&(N|3));
	else
	a = a >> 1;
	printf ("The shifted number is %u\n", a);
	return 0;
	}

11–15	#include<stdio.h>
	int f(int x){
	unsigned mask = 0x80000000;
	if (x & mask) return 1;
	else return 0;
	}
	int main () {
	printf ("The result is %d\n",f(N));
	printf ("The result is %d\n",f(-N));
	printf ("The result is %d\n",f(-2*N));
	printf ("The result is %d\n",f(2147483647));
	return 0;
	}

16–20	#include<stdio.h>
	int f(int x){
	unsigned mask = 0x1;
	return (x & mask);
	}
	int main () {
	printf ("The result is %d\n",f(N));
	printf ("The result is %d\n",f(2*N+1));
	printf ("The result is %d\n",f(2*N));
	return 0;
	}

21–25	#include<stdio.h>
	int f(int x){
	unsigned int mask = x >> 31;

Продовж. табл. 10.1

№	Фрагмент програми

	`return (x + mask)^ mask;`
	`}`
	`int main () {`
	`printf ("The result is %d\n",f(N));`
	`printf ("The result is %d\n",f(-N));`
	`printf ("The result is %d\n",f(0));`
	`return 0;`
	`}`

26–30	`#include<stdio.h>`
	`int f(int x){`
	`if (x && !(x & (x-1))) return 1;`
	`  else return 0;`
	`}`
	`int main () {`
	`printf ("The result is %d\n",f(N));`
	`printf ("The result is %d\n",f(2));`
	`printf ("The result is %d\n",f(4));`
	`printf ("The result is %d\n",f(6));`
	`printf ("The result is %d\n",f(32));`
	`return 0;`
	`}`

Завдання 10.2. Скласти програму знаходження наступних величин за варіантами, які наведені в таблиці 10.2. Задачі вирішувати за допомогою бінарних операцій.

Таблиця 10.2 – Варіанти завдання 10.2

№	Величини, які потрібно знайти
1–3	Дано ціле число A і натуральне число n. Виведіть число, яке складається тільки з n останніх біт числа A (тобто замініть на 0 всі біти числа A, крім останніх n). Результат навести у 2, 8, 16 та 10 системах числення.
4–6	Дано число $n<32$. Запишіть число 2^n, тобто число, у якого n-й біт дорівнює 1, а решта – нулі. Результат навести у 2, 8, 16 та 10 системах числення.

Продовж. табл. 10.2

№	Величини, які потрібно знайти
7–9	Дано ціле число A і натуральне число x. Запишіть нулі у останні x біт числа A його і виведіть результат. Результат навести у 2, 8, 16 та 10 системах числення.
10–12	Дано ціле число A і натуральне число x. Виведіть число, на яке перетвориться з число A, якщо встановити значення x-го біта рівному 1. Результат навести у 2, 8, 16 та 10 системах числення.
13–15	Дано два нерівних числа: n і m, що не перевищують 31. Обчисліть $2^n + 2^m$. Результат навести у 2, 8, 16 та 10 системах числення.
16–18	Дано ціле число A і натуральне число x. Виведіть число, на яке перетвориться число A, якщо інвертувати x-й біт. Результат навести у 2, 8, 16 та 10 системах числення.
19–21	Дано ціле число A, користуючись бінарними операціями, визначити знак числа (число є від'ємним чи ні). Число A навести у 2, 8, 16 та 10 системах числення.
22–24	Дано ціле число A і натуральне число x. Виведіть число, на яке перетвориться з число A, якщо встановити значення x-го біта рівному 0. Результат навести у 2, 8, 16 та 10 системах числення.
25–27	Дано ціле число A і натуральне число x. Треба отримати значення x-го біта числа A. Число A навести у 2, 8, 16 та 10 системах числення.
28–30	Дано ціле число A. Треба підрахувати кількість одиничних біт, що містить число A. Результат та число A навести у 2, 8, 16 та 10 системах числення.

Короткі теоретичні відомості

У мові С++ існує ряд операцій, що виконуються над розрядами. Вони носять назву бітові операції:

– унарна операція: інверсія бітів (~);

– бінарні операції: бітове "І" (&), бітове "АБО" (|), бітове виключне "АБО" (^), зсув вліво (<<), зсув вправо (>>).

Інверсія бітів (поразрядне заперечення) інвертує біти, тобто кожен біт зі значенням 1 отримує значення 0 і навпаки.

Бітове "І" порівнює послідовно розряд за розрядом два операнда. Для кожного розряду результат дорівнює 1, тоді і тільки тоді, коли обидва відповідні розряди операндів дорівнюють 1. Так, наприклад, `10010011 & 00111101 = 00010001`, тому що тільки нульовий і четвертий розряди обох операндів містять 1.

Бітове "АБО" порівнює послідовно розряд за розрядом два операнда. Для кожного розряду результат дорівнює 1 тоді і тільки тоді, коли будь-який з відповідних розрядів операндів рівні 1. Так, наприклад, `10010011 | 00111101 = 10111111`, тому що всі розряди (крім шостого) в одному з двох операндів мають значення 1.

Бітове виключне "АБО" порівнює послідовно розряд за розрядом два операнда. Для кожного розряду результат дорівнює 1, якщо один з двох (але не обидва) відповідних розрядів операндів дорівнює 1. Так, наприклад, `10010011 ^ 00111101 = 10101110`. Оскільки нульовий розряд в обох операндах має значення 1, нульовий розряд результату має значення 0.

Описані вище операції часто використовуються для встановлення деяких бітів, причому інші біти залишаються незмінними. Вони зручні для фільтрації або маскування бітів.

Зсув вліво зрушує розряди лівого операнда вліво на кількість позицій, визначених правим операндом. Звільнені позиції заповнюються нулями, а розряди, що зсуваються за ліву границю лівого операнду, губляться. Тому, наприклад, `10001010 << 2 = 00101000`, (кожен розряд зсунувся на дві позиції вліво).

Таким чином, `x<<2` зсуває x вліво на 2 позиції, заповнюючи звільнені позиції нулями (еквівалентно множенню на 4).

Зсув вправо пересуває розряди лівого операнду вправо на кількість позицій, визначених правим операндом. Звільнені позиції заповнюються нулями, а розряди, що зсуваються за праву границю лівого операнду, губляться. Для значень без знаку маємо `10001010 >> 2 = 00100010`, кожен розряд зсунувся на дві позиції вправо. Операції зсуву виконують ефективне множення і ділення на степені двійки.

Бітові поля в типових застосуваннях використовуються для зберігання цілих даних, частіше типу `unsigned`. Опис поля бітів складається з опису типу поля, його імені та зазначеного після двокрапки розміру поля в бітах, наприклад: `unsigned status: 6;`

Якщо ім'я поля відсутнє, то створюється приховане поле. Якщо розмір поля бітів представлений числом 0, то наступне поле бітів почнеться з границі машинного слова.

Приклад виконання роботи

Завдання 10.1. Визначити дію фрагмента програми

```
#include<stdio.h>
int f(int a, unsigned short int x){
return (a>>x)&1;}
int main () { int N=5;
printf ("The result is %d\n",f(N,0));
printf ("The result is %d\n",f(N,1));
printf ("The result is %d\n",f(N,2));
return 0;
}
```

Розв'язання

Цей фрагмент програми виводить на екран значення бітів, спочатку нульового, потім першого та другого.

```
The result is 1
The result is 0
The result is 1
```

Завдання 10.2. Скласти програму знаходження наступних величин: дано ціле число *a* та натуральне число *n*, користуючись бінарними операціями, визначити a^n.

Розв'язання

1. Постановка задачі.

Дано ціле число *a* та натуральне число *n*, користуючись бінарними операціями, визначити a^n.

Бінарне (двійкове) зведення до степеня – це прийом, що дозволяє зводити будь-яке число до *n*-го степеня за $O(log\ n)$ множень (замість *n* множень при звичайному підході).

2. Алгоритм розв'язання задачі.

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

Дія 1. Ввести значення чисел *a* та *n*.

Дія 2. Повторювати доки *n* не буде 0:

Дія 2.1. Якщо *n* непарне, то перейти до дії 2.2, інакше до дії 2.3;

Дія 2.2. Результату присвоїти результат помножити на число *a*; степінь *n* зменшити на 1;

Дія 2.3. Числу *a* присвоїти число *a* помножити на число *a*; степінь *n* поділити на 2;

Дія 3. Вивести результат на екран.

3. Текст програми

```
#include<stdio.h>
int binpow (int a, int n) {
     int res = 1;
     while (n)
          if (n & 1) {
               res *= a;
               --n;
          }
          else {
```

```
            a *= a;
            n >>= 1;
        }
    return res;
}
int main () {
int a;
unsigned int n;
printf ("Please enter an integer: ");
scanf ("%d", &a);
printf ("Please enter a power: ");
scanf ("%u", &n);
printf ("The result is : %d",binpow(a,n));
return 0;
}
```

4. Результати роботи програми

```
Please enter an integer: 6
Please enter a power: 3
The result is : 216
```

Контрольні питання

1. Що таке бітові операції?
2. Як працює операція інверсія бітів?
3. Як працює операція побітове "І"?
4. Як працює операція побітове "АБО"?
5. Як працює операція побітове виключне "АБО"?
6. Як працює операція зсув вліво?
7. Як працює операція зсув вправо?
8. Що таке бітове поле?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. **Буч, Г.** Объектно-ориентированное проектирование с примерами применения [Текст] : пер. с англ. / Г. Буч. – М. : Конкорд, 1992. – 519 с.

2. **Дьюхарт, С.** Программирование на Си++ [Текст] / С. Дьюхарт, К. Старк. – К. : НИПФ "ДиаСофт", 1993. – 272 с.

3. **Павловская Т. А.** C/C++. Программирование на языке высокого уровня. [Текст] / Т.А. Павловская. – СПб. : Питер, 2002. – 464 с.

4. **Прата, С.** Язык программирования C++. Лекции и упражнения, 6-е изд., обновл. и расшир. [Текст] / С. Прата. – М. : Диалектика – Вильямс, 2016. – 1248 с.

5. **Страуструп, Б.** Программирование: принципы и практика с использованием C++, 2-е изд.: Пер. с англ. [Текст] / Б. Страуструп – М. : ООО "И. Д. Вильямс", 2016. – 1328 с.

6. **Страуструп, Б.** Язык программирования C++ [Текст] / Б. Страуструп. – М. : Бином, 2004. – 369 с.

7. **Страуструп, Б.** Язык программирования C++. Специальное издание [Текст] / Б. Страуструп. – М. : Бином, 2011. – 1136 с.

8. **Шилдт, Г.** Самоучитель C++, 3-е издание [Текст] : пер. с англ. / Г. Шилдт. – СПб. : BHV – Санкт-Петербург, 1998. – 688 с.

9. **Шилдт, Г.** C++: базовый курс, 3-е издание [Текст] : пер. с англ. / Г. Шилдт. – М. : Издательский дом "Вильямс", 2010. – 624 с.

Додаток А

Функції для роботи з файлами з використанням структури FILE

Основні функції для роботи з файлами з використанням структури `FILE` наведені у таблиці А.1.

Таблиця А.1 – Основні функції для роботи з файлами з використанням структури `FILE`

Функція	Опис
`FILE *fopen (const char *fname, const char *mode)`	Відкриває файл, ім'я якого вказано аргументом `fname`, та повертає зв`язаний з ним вказівник. Тип операцій, дозволених над файлом, визначається аргументом `mode`. Дозволені для mode значення наведено в Додатку Б.
`size_t fread (void *buffer, size_t size, size_t count, FILE *stream)`	Повертає число прочитаних елементів. Дане значення може бути менше, ніж `count`, якщо було досягнуто кінець файлу або сталася помилка.
`size_t fwrite (const void *buffer, size_t size, size_t count, FILE *stream)`	Записує масив даних в файл. Повертає число записаних елементів `size`. Дане значення дорівнює `count`, якщо не сталася помилка.
`int fputs (const char * string, FILE *stream)`	Записує вміст рядка, на який вказує `string`, в заданий потік. Нуль в кінці рядка не записується. В разі успіху повертає останній записаний символ, а у разі невдачі – `EOF`.

Продовж. табл. А.1

Функція	Опис
`char *fgets (char *string, int n, FILE *stream)`	Зчитує до `n-1` символів із файлу `stream` та поміщає їх в масив символів, на який вказує `string`. Символи зчитуються до тих пір, доки не зустрінеться символ «новий рядок», `EOF` або до досягнення вказаної межі. По закінченню зчитування в масив `string` одразу після останнього зчитаного символу розміщується нульовий символ. Символ «новий рядок» при зчитуванні буде збережений та стане частиною масиву `string`.
`int fclose (FILE *stream)`	Використовується для закриття потоку, раніше відкритого за допомогою `fopen()`. Зберігає в файл дані, що знаходяться в дисковому буфері, та виконує операцію системного рівня по закриттю файлу. У разі успішного виконання повертає `0`, інакше – `EOF`.
`int fseek (FILE *stream, long offset, int origin)`	Встановлює вказівник положення в файлі, що зв`язаний зі `stream`, у відповідності зі значеннями `offset` та `origin`. Аргумент `offset` – виражений в байтах зсув від позиції, яка визначається `origin`, до нової позиції. Аргумент `origin` може приймати значення: 0 – початок файлу, 1 – поточна позиція, 2 – кінець файлу. У разі успіху повертає 0, ненульове значення означає невдачу.
`long ftell(FILE *stream)`	Повертає поточне значення вказівника положення у файлі для вказаного потоку. Це значення представляє собою кількість байт, на яку вказівник відстоїть від початку файлу. Повертає 1L у разі помилки. Якщо в даному потоці неможливий пошук за довільною адресою (в разі, наприклад, консолі), значення, що повертається, не визначається.

Продовж. табл. А.1

Функція	Опис
`int fflush(FILE *stream)`	Якщо `stream` зв`язаний з файлом, відкритим для запису, призводить до фізичного запису вмісту буфера в файл. Якщо `stream` зв`язаний з файлом, відкритим для читання, то очищується вхідний буфер. В обох випадках файл залишається відкритим. Повернення 0 означає успіх, а повернення ненульової величини вказує на наявність помилки по запису.
`int fileno (FILE *stream)`	Повертає дескриптор файлу вказаного потоку.

Додаток Б

Допустимі значення аргументу `mode` функції `fopen`

В якості типів доступу аргументу `mode` для використання в функції `fopen` можуть бути вказані наступні значення:

Значення	Опис
`r`	Файл відкривається тільки для читання. Якщо файл не існує, то функція генерує помилку (повертає 0).
`w`	Файл відкривається тільки для запису. Якщо файл не існує, то він буде созданий, якщо існує – початковий вміст буде перезаписано.
`a`	Файл відкривається для додавання в кінець (дозапису). Якщо файл не існує, то він буде созданий.
`r+`	Файл відкривається для читання та запису (файл повинен існувати).
`w+`	Файл відкривається для запису та читання. Якщо файл існує – початковий вміст буде перезаписано.
`a+`	Файл відкривається для додавання в кінець (дозапису) та читання. Якщо файл не існує, то він буде созданий.

Усі вищеописані типи доступу предназначено для текстового режиму відкриття файлу. Для двійкового режиму відкриття файлу після режиму достатньо добавити букву `b`. Наприклад, `rb`.

Додаток В

Команди форматування вводу/виводу для функцій `printf()` та `fprintf()` мови С

Таблиця В.1 – Перетворення, що використовуються у функціях `printf()` та `fprintf()`

Символ перетворення	Опис
`%`	Виводиться символ проценту.
`c`	Виводиться символ, що заданий відповідним аргументом.
`d`	Виводиться десяткове число зі знаком.
`e`	Значення типу `double` буде виведено в експоненційній формі.
`E`	Аналогічно попередньому, але в результат буде поміщена прописна (заголовна) буква `E`.
`f`	Значення типу `double` буде виведено в десятковому форматі.
`g`	Значення типу `double` буде виведено або в експоненційній формі, або в десятковому форматі. Автоматично обирається або перетворення `f`, або `e`, в залежності від того, яке з них дає більш точний результат в прийнятному невеликому полі виводу.
`G`	Аналогічно попередньому, але в разі представлення числа в експоненційній формі використовується прописна (заголовна) буква `E` замість рядкової (маленької) літери `e`.
`i`	Те ж саме, що і `d` (виводиться десяткове число зі знаком).
`n`	Значення аргументу інтерпретується як вказівник на змінну типу `int`, в якої функція запам`ятає число символів, записаних до цього на пристрій виводу. Це перетворення не відправляє на пристрій виводу ніяких символів.

Продовж. табл. В.1

Символ перетворення	Опис
o	Виводиться беззнакове вісімкове число.
p	Виводиться значення вказівника. В моделях пам'яті tiny (крихітної), small (малої) та medium (середньої) вказівники форматуються як шіснадцяткові значення зміщення. В моделях пам'яті compact (компактної), large (великої) та huge (огромної) вказівники форматуються як шіснадцяткові значення сегментного компоненту адреси та компоненту зміщення, розділені двокрапкою.
s	Виводиться вміст рядка, що завершується нулем. Задаючи значення точності, можна обмежити вивід максимальним числом символів, що вимагаються.
u	Виводиться беззнакове десяткове число.
x	Виводиться беззнакове шіснадцяткове число, для представлення якого використовуються цифри від 0 до 9 та строчні букви a, b, c, d, e, f.
X	Аналогічно попередньому, але використовуються прописні букви A, B, C, D, E, F.

Прапори задають правила вирівнювання, знаки "+" та "–", десяткову крапку, хвостові нулі та префікси для вісімкових та шіснадцяткових значень. Прапори не являються обов'язковими. Якщо вони присутні, то можуть складатися з одного або декількох символів.

Таблиця В.2 – Прапори, що використовуються у функціях printf() та fprintf()

Прапорець	Опис
-	Текст, що виводиться, вирівнюється по лівому краю. Порожній простір, що залишився справа, заповнюється пробілами. За замовчуванням текст вирівнюється по правому краю.

Продовж. табл. В.2

Прапорець	Опис
+	Числові значення випереджаються знаком плюса або мінуса. Зазвичай тільки від'ємні значення випереджаються знаком мінуса.
пробіл	Перед позитивними числовими значеннями виводиться пробіл, а перед від'ємними – знак мінуса. Цей режим діє за замовчуванням, тому не вживайте цей прапорець разом з прапорцем +. Не заключайте символ пробіла в апострофи.
#	У разі використання символу перетворення x або X ненульові аргументи випереджаються префіксом 0x або 0X відповідно. У разі використання символу o перетворення результат випереджаються цифрою 0. Якщо використовується перетворення e, E або f, у тексті, що виводиться та зображає число, буде присутня десяткова крапка (зазвичай вона виводиться тільки для дробних значень). Якщо використовується перетворення g або G, у тексті, що виводиться та зображає число, буде присутня десяткова крапка та хвостові нулі не подавляються (як це має місце за замовчуванням).

Додаток Г

Функції для роботи з файлами з використанням потоків мови С++

Основні функції для роботи з файлами з використанням потоків мови С++ наведені у таблиці Г.1.

Таблиця Г.1 – Основні функції для роботи з файлами з використанням потоків

Функція	Опис
`void open (const char* filename, ios_base::openmode mode = ios_base::in | ios_base::out);`	Відкриває файл, ім'я якого вказано аргументом `filename`, зв`язуючи його з об'єктом потоку. Аргумент `mode` визначає режим відкриття. Дозволені для mode значення наведено в Додатку Д. Якщо потік вже зв`язано з файлом (тобто він вже відкритий), виклик цієї функції завершується з помилкою.
`int istream::get();`	Читає єдиний символ із асоційованого потоку та поміщає його значення в `ch`. Повертає посилання на потік.
`ostream& ostream::put(char ch);`	Записує `ch` в потік та повертає посилання на цей потік.
`istream& istream::getline( char* buffer, streamsize num, char delim );`	Зчитує неформатовані дані з потоку в рядок. Зупиняється, коли знайдено символ, рівний роздільнику, або вичерпано потік. Перша версія використовує в якості роздільника `delim`, друга – `'\n'`. Символ-роздільник видаляється з потоку та не поміщається в рядок.
`istream& istream::read( char* buffer, streamsize num );`	Використовується з потоками вводу. Зчитує `num` байт із асоційованого потоку та посилає їх в буфер `buffer`. Якщо досягнутий кінець файлу `EOF`, зупиняється, залишаючи поточну кількість байт в буфері.

Продовж. табл. Г.1

Функція	Опис
`ostream& ostream::write( const char* buffer, streamsize num );`	Використовується з потоками виводу. Записує `num` байт в асоційований потік з буфера `buffer`.
`void close();`	Використовується для закриття файлу. Не має ні параметрів, ні значення, що повертається.
`bool istream::eof();`	При визові с дійсним дескриптором файлу повертає 1, якщо було досягнуто кінець файлу; в іншому випадку повертає 0. У разі помилки повертає -1.
`bool stream::fail();`	Повертає `true`, якщо виявлена помилка в поточному потоці, інакше повертає `false`. Функція може використовуватися для перевірки успішності попередньої операції.
`int gcount();`	Повертає число символів, прочитаних останнім оператором двійкового вводу.

Додаток Д

Допустимі значення аргументу mode методу open з вікористанням потоків мови С++

Відкрити файл в програмі при використанні потоків можна з використанням або конструкторів, або методу `open`, що має такі ж параметри, як і у відповідному конструкторі.

Значення аргументу mode при використання методу `open` є режим відкриття файлу. Замість значення за замовчуванням можна вказати одне з наступних значень, визначених в класі `ios`:

`ios::in`	Відкрити файл для читання (вибирається за замовчуванням для `ifstream`).
`ios::out`	Відкрити файл для запису (вибирається за замовчуванням для `ofstream`).
`ios::ate`	Встановити файловий вказівник на кінець файлу.
`ios::app`	Приєднувати дані; запис в кінець файлу.
`ios::trunc`	Знищити вміст, якщо файл існує (вибирається за замовчуванням, якщо прапор `out` вказаний, а прапори `ate` та `app` – ні).
`ios::binary`	Відкрити файл в двійковому режимі.
`ios::nocreate`	Якщо файл не існує, видати помилку, новий файл не відкривати.
`ios::noreplace`	Якщо файл існує, видати помилку, існуючий файл не відкривати.

Додаток Е

Директиви препроцесора

Основні директиви препроцесора наведені у таблиці Е.1.

Таблиця Е.1 – Директиви препроцесора

Директива	Опис
`#include <ім'я_файлу>`	Включення в вихідний текст програми вмісту файлу `им'я_файлу`. Препроцесор шукає файл в стандартних системних каталогах.
`#include "ім'я_файлу"`	Включення в вихідний текст програми вмісту файлу `им'я_файлу`. Спочатку препроцесор переглядає поточний каталог користувача, а потім звертається до стандартних системних каталогів.
`#define`	Використовується для визначення символічних констант і програмних конструкцій повністю.
`#undef`	Скасовує визначення, задане за допомогою `#define`.
`#if`	Директива умовної компіляції. Перевіряє значення константного цілочисельного виразу. Якщо воно відмінно від нуля, то вважається, що умова, яка перевіряється, істинна.
`#ifdef`	Директива умовної компіляції. Перевіряється, чи визначений за допомогою команди `#define` до поточного моменту ідентифікатор, уміщений після `#ifdef`.
`#ifndef`	Директива умовної компіляції. Перевіряється зворотна умова – істинним вважається невизначеність ідентифікатора, вміщеного після `#ifndef`, тобто той випадок, коли ідентифікатор не був використаний в команді `#define` або його визначення було скасовано командою `#undef`.

Продовж. табл. Е.1

Директива	Опис
`#else`	Директива умовної компіляції. За її наявності на компіляцію передається текст, наступний за директивою `#else`.
`#elif`	Директива умовної компіляції. Використовується для організації мультигалужень. Є скороченням конструкції `#else#if`.
`#endif`	Директива умовної компіляції. Визначає закінчення блоку команд умовної компіляції.
`#line <константа>`	Вказує компілятору, що наступний нижче рядок тексту має номер, який визначається цілою десятковою константою. Команда може визначати не тільки номер рядка, а й ім'я файлу.
`#error <послідовність_лексем>`	Призводить до видачі діагностичного повідомлення у вигляді послідовності лексем.
`#pragma <послідовність_лексем>`	Визначає дії, що залежать від конкретної реалізації компілятора, і дозволяє видавати компілятору різні інструкції.
`#`	Перетворює аргумент, якому передує, в рядок, взятий у лапки.
`##`	Використовується для конкатенації (об'єднання) двох лексем.

ЗМІСТ

Вступ 3
Робота № 1. Розробка та реалізація програм пошуку та сортування 4
Робота № 2. Розробка та реалізація програм для роботи з вказівниками та одновимірними динамічними масивами 14
Робота № 3. Розробка та реалізація програм з використанням багатовимірних динамічних масивів 25
Робота № 4. Розробка та реалізація програм для роботи зі структурами 36
Робота № 5. Розробка та реалізація програм з використанням форматування вводу/виводу, використовуючи консоль та текстовий файл 51
Робота № 6. Розробка та реалізація програм для роботи з бінарними файлами 65
Робота № 7. Розробка та реалізація програм з використанням директив препроцесора в додатках 76
Робота № 8. Розробка та реалізація програм для роботи з однонаправленними списками 87
Робота № 9. Розробка та реалізація програм з використанням рекурсивних алгоритмів 97
Робота № 10. Розробка та реалізація програм з використанням бітових операцій 106
Список рекомендованої літератури 114
Додаток А 115
Додаток Б 118
Додаток В 119
Додаток Г 122
Додаток Д. 124
Додаток Е 125