Лабораторна робота №1 Тема. Найпростіші класи та об'єкти

Теоретичне введення.Класи представляють абстрактні типи даних з відкритим інтерфейсом та прихованою внутрішньою реалізацією. У класах реалізовано базові засади *об'єктно- орієнтованого програмування*(ООП):

1) абстракція даних;

2) інкапсуляція – у класах поєднуються дані та методи (функції) до роботи з ними, тому що лише через методи можливий доступ до прихованих даних класу;

3) успадкування – у похідних класах успадковуються члени базового класу;

4) поліморфізм – можливість використання тих самих методів для роботи з різними об'єктами базового та породжених ним класів.

Визначення найпростішого класу без наслідування має вигляд:

```
class ім'я_класу {
   // за замовчуванням розділ private – приватні члени
   класу public: // відкриті функції та змінні класу };
```

Назва класу є новим типом даних. Поняття змінної цього типу відповідає поняття об'єкта класу. Список членів класу включає опис даних та функцій. Функції, описи яких у визначенні класу, називаються функціями- членами класу.

Змінні та функції, оголошені в розділі класу за промовчанням або явно як***private,***мають область видимості в межах класу. Їх можна зробити видимими поза класом, якщо оголосити у розділі ***public:***. Зазвичай змінні оголошуються у розділі***private***, а функції у розділі***public***.

Класами C++ є також структури***(struct)***та об'єднання ***(union*).**На відміну від класу, члени структури за умовчанням є відкритими, а чи не закритими. Крім того, об'єднання не можуть успадковуватись і успадковуватись.

При реалізації функціональної частини класу можна використовувати функції-члени класу, конструктори, деструктори, функції-оператори. Функції класу завжди оголошуються усередині класу. Визначення функції може бути і всередині класу. Такі функції називаються***inline***-функціями. Зазвичай визначення

функцій-членів класу розміщуються поза класом. При цьому перед ім'ям функції вказується ***ім'я_класу::***.

тип ім'я_класу: ім'я_функції (опис аргументів)

{ /*тіло функції*/ }

Виклик функцій здійснюється одним із двох способів:

ім'я_об'єкта.ім'я_функції (аргументи);

вказівник_на_об'єкт -> ім'я_функції (аргументи);

Звернення до даних об'єкта класу здійснюється за допомогою функцій, що викликаються з об'єктів. У цьому функції-члену класу передається прихований покажчик ***this*** на об'єкт, що викликає функцію.

Функції-«друзі» класу, що оголошуються у класі зі специфікатором ***friend***, покажчик ***this*** не містять. Об'єкти, з котоними працюють такі функції, повинні передаватися як їх аргументи. Це звичайні функції мови С++, яким дозволено доступом до закритим членам класу.

приклад.

```
/*Створюється класСтудент. Формується динамічний масив об'єктів. При
тестуванні виводиться: сформований список студентів, список студентів
заданого факультету, список студентів для заданих факультету та курсу.*/
# include <conio.h>
# include <string.h>
# include <iostream.h>
struct date // дата народження
  {char daymon[6];
   int year; };
//======= class Student =================
   class Student{
   char name [30]; //private
   date t;
   char adr [30], fac [20];
   int kurs;
     public:
       Student();
       char * getfac ();
       int getkurs();
       void show();
        };
Student::Student()
       {cout<<"Input name:"; cin>>name;
       cout<<"Input date of born\n";
       cout<<"Day.mon:";       cin>>t.daymon;
       cout<<"Year:";         cin>>t.year;
       cout<<"Input adr:"; cin>>adr;
       cout<<"Input fac:"; cin>>fac;
```

```
      cout<<"Input kurs:"; cin>>kurs;
}
Student::show()
   {
    cout<<"Name       :"<<name<<endl;
    cout<<"Was born :"<<t.daymon<<'.'<<t.year<<endl;
    cout<<"Address     :"<<adr<<endl;
    cout<<"Fac        :"<<fac<<endl;
    cout<<"Kurs       :"<<kurs<<endl;
    }
char *Student::getfac() { return fac; }
int Student::getkurs() { return kurs; }
void spisfac(Student spis[],int n)//список студентів заданого факультету
    {char fac[20];
     cout<<"Input faculty:"; cin>>fac;
     for(int i=0;i<n;i++)
         if(strcmp(spis[i].getfac(),fac)==0)spis[i].show();
    }
void spisfackurs(Student spis[],int n)
//список студентів заданих факультету та курсу
    {int i,k;
     char fac [20];
     cout<<"Input faculty:"; cin>>fac;
     cout<<"Input the course:"; cin>>k;
     for(i=0;i<n;i++)
     if ((strcmp(spis[i].getfac(),fac)==0)&&(spis[i].getkurs()==k))
            spis[i].show();
    }
//========= main ================ void
main()
{ Student * spis;
    int n;
    cout<<"Input a number of students: "; cin>>n;
    spis=new Student [n];
    for(int i=0;i<n;i++) {
      cout<<"============================="<<endl;
      spis[i].show();
      }
spisfac(spis,n);
spisfackurs(spis,n);
delete [] spis;
cout<<"press any key!"
while(!kbhit());
}
```

Завдання для самостійного вирішення

Розробити класи для описаних нижче об'єктів. Включити до класу методи set (...), get (...), show (...). Визначити інші способи.

1.**Student**: Прізвище, Ім'я, По батькові, Дата народження, Адреса, Телефон, Факультет, Курс. Створити масив об'єктів. Вивести:

а) перелік студентів заданого факультету;

б) списки студентів для кожного факультету та курсу; в) список студентів, які народилися після цього року.

2.**Abiturient**: Прізвище, Ім'я, По-батькові, Адреса, Оцінки. Створити масив об'єктів. Вивести:

а) список абітурієнтів, які мають незадовільні оцінки; б) список абітурієнтів, сума балів у яких не менша за задану;

в) вибрати N абітурієнтів, які мають найвищу суму балів, та список абітурієнтів, які мають півпрохідний бал.

3.**Aeroflot**: Пункт призначення, Номер рейсу, Тип літака, Час. вильоту, дні тижня. Створити масив об'єктів. Вивести:

а) список рейсів для заданого пункту призначення;

б) список рейсів для заданого дня тижня;

в) список рейсів для заданого дня тижня, час вильоту для яких більший за заданий.

4.**Book**: Автор, Назва, Видавництво, Рік, Кількість сторінок. Створити масив об'єктів. Вивести:

а) перелік книг заданого автора;

б) перелік книжок, випущених заданим видавництвом; в) список книг, що випущені після заданого року.

5.**Worker**: Прізвище та ініціали, Посада, Рік надходження на роботу, Зарплата. Створити масив об'єктів. Вивести:

а) список працівників, стаж роботи яких на даному підприємстві перевищує задану кількість років;

б) список працівників, зарплата яких більша за задану; в) список працівників, які обіймають задану посаду.

6.**Train**: Пункт призначення, Номер поїзда, Час відправлення, Число загальних місць, купейних, плацкартних. Створити масив об'єктів. Вивести:

а) список поїздів, що прямують до заданого пункту призначення; б) список поїздів, що прямують до заданого пункту призначення та відправляються після заданої години;

в) список поїздів, що вирушають до заданого пункту призначення та мають спільні місця.

7.**Product**: Назва, Виробник, Ціна, Термін зберігання, Кількість. Створити масив об'єктів. Вивести:

а) перелік товарів для заданого найменування;

б) перелік товарів для заданого найменування, вартість яких не перевищує зазначеної;

в) список товарів, термін зберігання яких більший за заданий. 8. **Patient**: Прізвище, Ім'я, По-батькові, Адреса, Номер медичної карти, діагноз. Створити масив об'єктів. Вивести:

а) список пацієнтів, які мають цей діагноз;

б) список пацієнтів, номер медичної картки яких перебуває у заданому інтервалі.

9.**Bus**: Прізвище та ініціали водія, Номер автобуса, Номер маршруту, Марка, Рік початку експлуатації, Пробіг. Створити масив об'єктів. Вивести:

а) список автобусів для заданого номера маршруту;

б) список автобусів, які експлуатуються понад 10 років; в) список автобусів, пробіг яких більше 10 000 км.

10.**Customer**: Прізвище, Ім'я, По-батькові, Адреса, Телефон, Номер кредитної картки, номер банківського рахунку. Створити масив об'єктів. Вивести:

а) список покупців у алфавітному порядку;

б) список покупців, номер кредитної картки яких перебуває у заданому інтервалі.

11.**File**: Ім'я файлу, Розмір, Дата створення, Кількість оберне-ній. Створити масив об'єктів. Вивести:

а) список файлів, упорядкований у алфавітному порядку;

б) список файлів, розмір яких перевищує заданий;

в) список файлів, кількість звернень до яких перевищує задане.

12.**Word**: Слово, Номери сторінок, на яких слово зустрічається (від 1 до 10), Кількість сторінок. Створити масив об'єктів. Вивести:

а) слова, що зустрічаються більш ніж на сторінках N; б) слова за абеткою;

в) для заданого слова номера сторінок, на яких воно зустрічається.

13.**House**: Адреса, Поверх, Кількість кімнат, Площа. Створити масив об'єктів. Вивести:

а) список квартир, що мають задану кількість кімнат;

б) список квартир, що мають задану кількість кімнат і розташовані на поверсі, що знаходиться в певному проміжку;

в) список квартир, що мають площу, що перевищує задану. 14.

Phone: Прізвище, Ім'я, По-батькові, Адреса, Номер, Час всередині-міських розмов, Час міжміських розмов. Створити масив об'єктів. Вивести:

а) відомості про абонентів, час внутрішньоміських розмов яких перевищує заданий;

б) відомості про абонентів, які скористалися міжміським зв'язком;

в) відомості про абонентів, які виведені в алфавітному порядку. 15.

Person: Прізвище, Ім'я, По батькові, Адреса, Стать, Освіта, Рік народження. Створити масив об'єктів. Вивести:

а) перелік громадян, вік яких перевищує заданий; б) список громадян із вищою освітою;

в) перелік громадян чоловічої статі.

Тести

1. Навіщо потрібні класи?

Варіанти відповіді:*

* 1) для визначення нових типів у програмі; 2) для спрощення роботи з функціями; *3) для з'єднання даних та операцій над ними; 4) для спрощення роботи з покажчиками.

2. Методи класу визначають:

* 1) які операції можна виконувати з об'єктами цього класу; 2) які типи значень можуть набувати дані-члени класу; 3) як реалізується захист даних-членів класу.

3. Атрибути (дані-члени) класу можуть бути:

1) лише цілими числами; 2) будь-якими вбудованими типами; *3) будь-якого визначеного в програмі типу.

4. Який із записів відповідає зверненню до атрибуту arg класу A в визна-розподілі методу цього ж класу?

Варіанти відповіді:

* 1) this -> arg; *2) arg; 3) A -> arg; 4) A -> this -> arg.

5. Якщо є код

class A {public: int a; }; A * obj;

то як звернутися до змінної a?

Варіанти відповіді:

1) obj.a; 2) (*obj) -> a; *3) obj -> a; 5) obj :: a.

6. Якщо є код

class A {public: int a, b, c;}; A * obj;

то як звернутися до змінної b?

Варіанти відповіді:

1) (*obj) -> b; 2) A:: b; *3) (*obj).b; 4) obj -> ab

---

* Зірочкою у тестах відзначені правильні відповіді.

7. Яким буде результат виконання наступного коду:

```
class A {
public:
int inc (int x) {return x++;} int
inc (short x) (return x+2;}};

A obj; int y=5;
cout << obj.inc(y); ?
```

Варіанти відповіді:

* 1) 5; 2) 6; 3) 7; 4) помилка під час компіляції.

8. Яким буде результат виконання наступного коду:

```
class A {
public:
int y;
int inc (int x) {return y++;} int
inc (short x) {return x+y;}};

A obj; int y=5; obj.y = 6;
cout << obj.inc(y); ?
```

Варіанти відповіді:

1) 5; *2) 6; 3) 11; 4) 7; 5) помилка під час компіляції.

9. Якими за промовчанням оголошуються елементи структури?

Варіанти відповіді:

1) private; *2) public; 3) protected; 4) за умовчанням не оголошуються.

Лабораторна робота №2
Тема. Розробка класів

Теоретичне введення.При розробці класу необхідно оп-поділити дані класу та його методи, конструктори та деструктори. Конструктор – це функція-член класу, яка викликається автоматично під час створення статичного чи динамічного об'єкта класу. Він ініціалізує об'єкт та змінні класу. У конструктора немає значення, що повертається, але він може мати аргументи і бути перевантажуваним.

Протилежні конструктору дії виконує деструктор, який викликається автоматично при знищенні об'єкта. Деструктор має те саме ім'я, що й клас, але перед ним стоїть '~'. Деструктор можна викликати явно на відміну конструктора. Конструктори та деструктори не успадковуються, хоча похідний клас може викликати конструктор базового класу.

*Оператори-функції.*Використовуються для введення операцій над об'єктами, що зв'язуються із символами:

+ , - , * , / , % , , & , | , ~ , ! , = , < , > ,+= , [] , -> , ( ) , new, delete.

Оператор-функція є членом класу або дружньої (***friend***) класу. Загальна форма оператор-функції-члена класу:

повертається_тип
***ім'я_класу*::operator#(список_аргум)**
{/*тіло функції*/}

Після цього замість ***operator#(a,b)*** можна писати ***a#b***. Тут # представляє один із введених вище символів. Прикладами є оператори ***>>*** і ***<<*** – оператори введення-виведення, що перевантажуються. Зазначимо, що під час перевантаження не можна змінювати пріоритет операторів і кількість операндів. Якщо оператор-функція-член класу перевантажує бінарний оператор, то функція матиме лише один параметр-об'єкт, що стоїть праворуч від знака оператора. Об'єкт зліва викликає оператор-функцію і передається неявно за допомогою покажчика ***this***. До дружньої функції покажчик ***this*** не передається, тому унарний оператор має один параметр, а бінарний – два.

Оператор присвоювання може бути дружньою функцією, лише членом класу.

приклад. Створюється клас Поліном. У головній програмі виконується тестування класу.

```
# include <iostream.h>
# include <conio.h>
# include <math.h>
class Polynom {
 int n;
 double *koef;
 public:
 Polynom(); //Конструктори
 Polynom(int k);
 Polynom (int k, double * mas);
 Polynom(const Polynom&ob); //конструктор копіювання
 ~Polynom(){delete[]koef;}
 void GiveMemory(int k);
 void SetPolynom(int k,double *mas);
 void SetDegree(int k) {n=k;}; //Встановити ступінь void
 CalculateValue(double x); //обчислити значення int
 GetDegree(){return n;}; //отримати ступінь double
 GetOneCoefficient(int i){return(koef[i]);}; Polynom operator+
 (Polynom ob); //перевантаження операторів Polynom
 operator*(Polynom ob);
 double& operator[](int i){return(koef[i]);}//перевантаження []
 Polynom& operator = (const Polynom p) {
   if(&p==this) return *this;
   if(koef) delete [] koef;
```

```
    n=pn;
    koef=new double [p.n+1];
    for(int i=0;i<=pn;i++)
        koef[i]=p.koef[i];
    return *this;
}
 friend ostream& operator<<(ostream& mystream,Polynom &ob);
 friend istream& operator>>(istream& mystream,Polynom &ob); int min
(int n, int m)
{return(n<m)? n: m; } int
 max (int n, int m)
 {return (n> m)? n: m; } };

//********** Polynom() **********************************
Polynom::Polynom()
{randomize();
 n=random(5);
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=n;i>=0;i--)
     koef[i]=random(10)-5;
}
//************* Polynom(int k) ****************************** *
Polynom::Polynom(int k)
{n=k;
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=n;i>=0;i--)
     koef[i]=random(10)-5;
}
//****************** Polynom(int k,double mas[]) ******************
Polynom: :Polynom(int k,double mas[])
{n=k;
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=n;i>=0;i--)
     koef[i]=mas[i];
}
//*************** Polynom(const Polynom&ob) **********************
Polynom::Polynom(const Polynom&ob) )
{n=ob.n;
 koef=new double[n+1];
 if(!koef){cout<<"Error";getch();return;}
 for(int i=0;i<=n;i++)
    koef[i]=ob.koef[i];
}
//***************** void GiveMemory(int k) **********************
```

```
void Polynom::GiveMemory(int k) {

 if(koef) delete [] koef;
 koef=new double[k+1];
 if(!koef){cout<<"Error";getch();return;} }

//******************** SetPolynom ************************** void
Polynom::SetPolynom(int k,double *mas)
{n=k;
 if(koef) delete [] koef; koef
 = new double [n+1]; for(int
 i=n;i>=0;i--)
     koef[i]=mas[i];
}
//*************** CalculateValue ***************************** void
Polynom: :CalculateValue(double x=1.0)
{ double s;
  int i;
 for(s=koef[0],i=1;i<=n;i++)
    s=s+koef[i]*pow(x,i);
 cout<<"f("<<x<<")="; cout<<s<<endl; }

//***************** Polynom operator+(Polynom ob) ***************
Polynom Polynom::operator+(Polynom ob)
{ int i;
 Polynom rab;
 rab.GiveMemory(max(n,ob.GetDegree()));
 for(i=0;i<=min(n,ob.GetDegree());i++)
    rab.koef[i]=koef[i]+ob.GetOneCoefficient(i);
    if(n<ob.GetDegree())
   {
    for(i=min(n,ob.GetDegree())+1;i<=ob.GetDegree();i++)
       rab.koef[i]=ob.GetOneCoefficient(i);
       rab.n=ob.GetDegree();
   }
 else
   {
    for(i=min(n,ob.GetDegree())+1;i<=n;i++) rab.koef[i]=koef[i];
    rab.n=n;
   }
 return rab;
}
//*************** Polynom operator*(Polynom ob) ***************
Polynom Polynom::operator*(Polynom ob)
{
 int i,j,k;
 double s;
```

```
 Polynom rab;
 rab.GiveMemory(n+ob.GetDegree());
 for(i=0;i<=n+ob.GetDegree();i++)
    {s=0;
     for(j=0;j<=n;j++)
        for(k=0;k<=ob.GetDegree();k++)
           if(j+k==i)s=s+koef[j]*ob.GetOneCoefficient(k);
           rab.koef[i]=s;
    }
 rab.n=n+ob.GetDegree();
 return rab;
}
//********** ostream& operator<<(ostream& mystream,Polynom &ob) ******
ostream& operator<<(ostream& mystream,Polynom &ob)
{char c=''; //пропустимо "+" перед першим коефіцієнтом
   for(int i=ob.n;i>=0;i--)
    { double ai=ob.koef[i];
         if(ai==0) continue;
            else {if(ai>0) mystream<<c; mystream<<ai;}
          if(i==0) continue; else mystream<<"x"; if(i==1)
          continue; else mystream<<"^"<<i; if(ai!=0)c='+';

    }
 if(c==' ')mystream<<0;
 mystream<<endl;
 return mystream;
}
//********* istream& operator>>(istream& mystream,Polynom &ob) *
istream& operator>>(istream& mystream,Polynom &ob)
{
 int i;
 cout<<"Enter Degree:"; mystream>>ob.n; cout<<endl;
 for(i=ob.n;i>=0;i--)
    {
     cout<<"Enter koeff "<<i<<":"; mystream>>ob.koef[i]; }

 return mystream;
}
//******************** MAIN ************************** * int
main (int argc, char * argv [])
{const int m=3;
   Polynom f, g, masp [m], * p1, s;
   int n=5,i;
   double K[6]={1.0,3.2,0.0,4.1,0.0,1.1};
    p1=new Polynom(n,K);
    cout << p1;
    p1->CalculateValue(2.0);
```

```
  cin>>f;
     cout<<" f(x)= "; cout<<f;
    cout<<" g(x)= "; cout<<g; s =
 f + g;
 cout<<"f(x)+g(x) = "; cout<<s; s =
 f * g;
 cout<<" f(x)*g(x) = "; cout<<s;
 s=masp[0]; cout << masp [0];
 for(i=1;i<m;i++)
   { s = s + masp [i]; cout<<masp[i];}
 cout<<"Summa: "; cout<< s; while(!
 kbhit());
 delete p1;
 return 0;
}
```

Завдання для самостійного вирішення

Розробити наведені нижче класи. При розробці кожного класу можливі два варіанти рішення: а) дані-члени класу є змінними та масивами фіксованої розмірності; б) пам'ять даних-членів класу виділяється динамічно.

1. «**Комплексне число**» –**Complex**. Клас повинен містити кілька конструкторів та операції для складання, віднімання, множення, поділу, присвоювання. Створити два вектори розмірності *n* з комплексних координат. Передати їх у функцію, яка виконує складання комплексних векторів.

2. Визначити клас «**Дріб**» –**Fraction** у вигляді пари (m,n). Клас має містити кілька конструкторів. Реалізувати методи для складання, віднімання, множення та поділу дробів. Перевантажити операції додавання, віднімання, множення, поділу, привласнення та операції відношення. Створити масив об'єктів та передати його у функцію, яка змінює кожен елемент масиву з парним індексом шляхом додавання наступного за ним елемента масиву.

3. Розробити клас «**Вектор**» –**Vector** розмірності *n*. Визначити кілька конструкторів, у тому числі конструктор копіювання. Реалізувати методи для обчислення модуля вектора, скалярного твору, додавання, віднімання, множення на константу. Перевантажити операції додавання, віднімання, множення, інкременту, декременту, індексування, присвоювання для даного класу. Створити масив об'єктів. Написати функцію, яка для заданої пари векторів визначатиме, чи є вони колінеарними чи ортогональними.

4. Визначити клас «**Квадратна матриця**» –**Matrix**. Клас долдружин містити кілька конструкторів, у тому числі конструктор копіювання. Реалізувати методи додавання, віднімання, множення матриць; обчислення норми матриці. Перевантажити операції складення, віднімання, множення та присвоювання для даного класу. Створити масив об'єктів класу**Matrix**і передати його в функцію, котораю зраджує*i*-ю матрицю шляхом зведення її у квадрат. У головній програмі вивести результат.

5. Розробити клас**«Многочлен»Polynom**ступеня*n*. Написати кілька конструкторів, у тому числі конструктор копіювання. реалізувати методи для обчислення значення полінома; складання, віднімання та множення поліномів. Перевантажити операції додавання, віднімання, множення, інкременту, декременту, індексування, присвоєння. Створити масив об'єктів класу. Передати його у функцію, яка обчислює суму поліномів масиву та повертає поліном-результат, який виводиться на екран у головній програмі.

6. Визначити клас**"Стек" -Stack**. Елементи стека зберігаються в масиві. Якщо масив має фіксовану розмірність, то передбачити контроль виходу за межі масиву. Якщо пам'ять виділяється динамічно і не вистачає, то збільшити розмір виділеної пам'яті. Включення елементів у стек та їх вилучення реалізувати як у вигляді методів, так і за допомогою перевантажених операцій. Створити масив об'єктів класу**Stack**. Передавати об'єкти у функцію, яка видаляє зі стека перший (згори), третій, п'ятий тощо елементи.

7. Побудувати класи для опису плоских фігур: коло, квадрат, прямокутник. Включити методи зміни об'єктів, переміщення на площині, обертання. Перевантажити операції, що реалізують самі дії. Виконати тестування класу, створивши масив об'єктів.

8. Визначити клас «**Рядок**» –**String**довжини*n*. Написати нескільки конструкторів, зокрема конструктор копіювання. Реалізувати методи для виконання конкатенації рядків, вилучення символу із заданої позиції, порівняння рядків. Перевантажити операції додавання, індексування, відносини, додавання (+ =) , присвоєння для даного класу. Створити масив об'єктів і передати їх у функцію, яка виконує сортування рядків.

9. Розробити клас «**Безліч (цілих чисел, символів, рядків** і т.д.)" -**Set**потужності*n*. Написати кілька конструкторів, зокрема конструктор копіювання. Реалізувати методи для визначення належності заданого елемента безлічі, перетину, об'є-

діння, різниці двох множин. Перевантажити операції додавання, віднімання, множення (перетину), індексування, привласнення. Створити масив об'єктів і передавати пари об'єктів у функцію, яка будує безліч, що складається з елементів, що входять лише в одну із заданих множин, тобто.$(A \cup B) \setminus (A \cap B)$, і повертає його в головну програму.

10. Розробити клас для масиву рядків. Написати кілька конструкторів, у тому числі конструктор копіювання. Реалізувати методи для поелементної конкатенації двох масивів, упорядкування рядків у лексикографічному порядку, злиття двох масивів з видаленням рядків, що повторюються, а також для виведення на екран всього масиву та заданого рядка. Перевантажити операції додавання, множення, індексування, присвоєння даного класу. Створити масив об'єктів та передавати об'єкти у функцію, яка виконує злиття об'єктів і для отриманого об'єкта-результату здійснює лексикографічне впорядкування рядків.

11. Скласти опис класу, що забезпечує подання матриці заданого розміру$n \times m$та будь-якого мінору в ній. Пам'ять для матриці виділяти динамічно. Написати кілька конструкторів, зокрема конструктор копіювання. Реалізувати методи для відображення на екрані як матриці в цілому, так і заданого мінору, а також зміни мінору; додавання, віднімання, множення мінорів. Перевантажити операції додавання, віднімання, множення та присвоювання для даного класу. Створити масив об'єктів даного класу та передати його у функцію, яка змінює дляi-ї матриці її мінор шляхом множення на константу.

12. Побудувати клас «**Бульов вектор**»-**BoolVector**розмірностіn. Визначити кілька конструкторів, у тому числі конструктор копіювання. Реалізувати методи для виконання порозрядних кон'юнкцій, диз'юнкції та заперечення векторів, а також підрахунку числа одиниць та нулів у векторі. Реалізувати самі дії над векторами за допомогою перевантажених операцій. Перевантажити операції відношення та присвоювання для даного класу. Створити масив об'єктів. Передавати об'єкти у функцію, яка їх змінюватиме за формулою $A = A \vee \overline{B}$.

13. Реалізувати клас «**Трійковий вектор**»-**Tvector**розмірності n. Компоненти вектора приймають значення з множини$\{0, 1, X\}$. Два трійкові вектори$t_k = (t_{k1}, \ldots, t_{kn})$та$t_l = (t_{l1}, \ldots, t_{ln})$називаються ортогональними, якщо існує такеi, що$t_{ki}, t_{li} \in \{0,1\}$та$t_{ki} \neq t_{li}$. Операція

перетину не ортогональних векторів виконується покомпонентно за такими правилами: $1\cap 1=1\cap X=X\cap 1 = 1$, $0\cap 0=0\cap X=X\cap 0=0$, $X\cap X=X$. Реалізувати методи перевірки векторів на ортогональність, для перетину не ортогональних векторів, порівняння векторів, підрахунку числа компонент, рівних X. Здійснити ті ж дії над векторами за допомогою перевантажених операцій. Перевантажити операцію надання для даного класу. Виконати тестування класу, створивши масив об'єктів.

14. Визначити клас «**Булева матриця»**-**BoolMatrix**розмірності $n\times m$. Клас повинен містити кілька конструкторів, зокрема конструктор копіювання. Реалізувати методи для логічного додавання (диз'юнкції), множення та інверсії матриць. Реалізувати методи для підрахунку числа одиниць у матриці та лексикографічного впорядкування рядків. Перевантажити операції для логічного складу, множення та інверсії матриць, а також операцію присвоєння. Створити масив об'єктів класу**BoolMatrix.**Передавати об'єкти в функцію, яка їх змінює за формулою $A=A\vee B$.

Тести

1. У якому рядку допущено помилку?

```
class X {
   int a;
   int f() const;            //1
   int g() {return a++;}        //2
   int h() const {return a++;} //3
};
```

Варіанти відповіді:

1) тут немає помилок; (*) 2) //3; 3) //2; 4) //1.

2. Виберіть три правильні твердження:

[1] статична функція-член класу, що не має покажчик цього;
[2] статична функція-член класу може бути віртуальною;
[3] в класі не може бути двох функцій: статичної та нестатичної – з одним і тим самим ім'ям та списком параметрів;
[4] статична функція у класі може бути оголошена const.

Варіанти відповіді:

1) усі твердження не вірні; (*) 2) [4]; (*) 3) [3]; 4) [2]; (*) 5) [1]. 3 Чи правильно перевантажені функції?

```
class X {
   static void f();
   void f() const;
};
```

Варіанти відповіді:

* 1) ні; 2) так; 3) так, якщо прибрати const.

4. Які з операторів не можуть бути перевантажені?

Варіанти відповіді:

1) new; *2) . ; 3) *; 4) []; 5) ().

5. Що буде виведено?

```
# include < iostream.h >
class X {
   int a;
public:
   X() : a(1) {}
   X& operator++() {a++; return *this;}; X&
   operator++(int) {a--; return *this;}; friend
   ostream& operator<<(ostream&, X&);
};
ostream& operator<<(ostream& _stream, X& _x) {
   _stream << _x.a;
   return _stream;
}
void main() {
   X x;
   cout << ++x;
   cout << x++;
}
```

Варіанти відповіді:

1) нічого; *2) 21; 3) 12; 4) 22; 5) 11.

6. Дано такі три властивості, якими володіє функція-член класу:

[1] функція має право доступу до закритої частини оголошення класу;

[2] функція знаходиться в області видимості класу;

[3] функція повинна викликатися для об'єкта класу (є покажчик цього). Якими з цих властивостей має дружня функція?

Варіанти відповіді:

1) [3]; 2) жодним; 3) [2]; *4) [1].

7. Які функції неявно inline? Виберіть три варіанти відповіді.

[1] всі дружні функції;

[2] дружні функції, визначені у класі;

[3] неявний конструктор за замовчуванням;

[4] неявний деструктор за умовчанням;

[5] Віртуальні функції.

Варіанти відповіді:

1) [5]; *2) [4]; *3) [3]; *4) [2]; 5) [1].

8. Даний наступний код:

```
class X {
   friend void f(X&);
};
```

За допомогою якого синтаксису може бути спричинена функція f?

Варіанти відповіді:

1) f не може бути викликана, тому що вона оголошена як private; 2) X x; X::f(x); 3) X x; f(&x); 4) X x; xf(x); *5) X x; f(x).