ЗАДАНИЕ 1. ПРОСТЕЙШИЕ КЛАССЫ И ОБЪЕКТЫ

Цель задания **–** изучение приёмов объектного программирования с использованием классов, объектов, конструкторов, деструкторов.

Основные понятия

Класс представляет собой тип, определяемый пользователем, и в простейшем случае имеет вид:

```
class имя_класа{
 //члены класса
};
```

Определение класса соответствует введению нового типа данных, а объявление переменной данного типа создаёт объект этого класса. Список членов класса включает данные, представляющие объект, и функции, выполняющие действия над этими данными.

Объект – это экземпляр класса. Класс задаёт целую категорию объектов. Все объекты имеют общие функции-члены, но каждый объект создаёт и поддерживает свои собственные данные-члены.

По умолчанию все данные-члены и функции-члены класса являются закрытыми (***private***), т.е. доступны только в пределах класса. Чтобы сделать их доступными вне класса, используется модификатор ***public***. Обычно данные класса объявляются как ***private,*** а функции – ***public***:

```
class Samp{
  int n, m;      // private-члены
public:
  init();        // public-члены
};
```

Определение класса только задаёт тип объекта, но не создаёт ни одного объекта, следовательно не производит выделения памяти под объект. Для создания объекта испольуют имя класса как спецификатор типа данных:

```
Samp ob1, ob2;      // создание объектов ob1 и ob2 типа Samp
```

После создания объекта класса может возникнуть необходимость в получении доступа к членам этого объекта. Открытые функции-члены класса имеют доступ ко всем членам своего класса, в том числе и к закрытым, через них возможен доступ к этим членам. К открытым членам класса можно обращаться, используя имя объекта и *операцию точка* **(.):**

```
ob1.init();     // вызов функции init() для объекта ob1
ob2.init();     // вызов функции init() для объекта ob2
```

Функции-члены класса определяют как обычные функции. Но для того чтобы показать, что эти функции принадлежат классу, перед именем функции пишется имя класса и *операция разрешения области видимости* **(::)**. Например:

```
Samp::init(){          // определение функции init() класса Samp
  n = 10;     m = 20;
}
```

Определение функции может находиться и внутри класса. Такие функции называются ***inline***-функциями.

Возможны ситуации, когда для получения доступа к закрытым членам класса будет нужна функция, не являющаяся членом этого класса. В этом случае следует использовать **дружественную функцию**, которая не является членом класса, но имеет доступ к закрытым членам класса (только через объект**).**

Дружественная функция задаётся и вызывается как обычная функция. Однако в определение класса, для которого функция будет дружественной, необходимо включить её прототип, перед которым поставить ключевое слово ***friend***. Дружественная функция может быть дружественной более чем одному классу, может быть членом одного класса и дружественной другому.

Конструктором называется функция-член класса, которая автоматически вызывается при объявлении (т.е. создании) объекта и имя которой совпадает с именем этого класса. Конструктор должен быть открытым членом класса, может быть перегружен, может иметь аргументы по умолчанию; ему можно передать аргументы, но он не должен возвращать значения. Конструктор используется для инициализации данных-членов класса.

Конструктор по умолчанию – это конструктор, который совсем не имеет параметров или у которого все параметры заданы как аргументы по умолчанию. Класс может иметь ***только один конструктор по умолчанию***. Если класс не имеет никакого конструктора, то компилятор, если надо, сгенерирует неявный конструктор по умолчанию сам.

Деструктор – это функция-член класса, имя которой совпадает с именем этого класса, перед которым дополнительно ставится символ “~” (тильда). Класс может иметь только один деструктор или ни одного, тогда он будет вызван по умолчанию. Деструктор не может иметь параметры и не должен иметь возвращаемого значения. Деструктор автоматически вызывается при удалении объекта из памяти. Использовать деструктор имеет смысл в случае, если объект требует выделения динамической памяти.

Постановка задания

Определить класс для решения задач вычислительного характера, предусмотрев в нём функции для инициализации, ввода, вывода данных класса, функции для доступа к членам класса и для работы с объектами данного класса. Написать программу, которая демонстрирует использование созданного класса для работы с натуральными числами.

Примеры выполнения задания

// **Пример 1.1.** Нахождение суммы, произведения двух целых чисел.
// Использование конструктора.

```
#include <iostream.h>
class Chisla{                  // определение класса Chisla
  int a;                       // число a
  int b;                       // число b
public:
  Chisla(int x, int y);        // конструктор с параметрами
  int sum();                   // функция получения суммы чисел
  int mult();                  // функция получения произведения чисел
  show();                      // функция вывода чисел
};
  Chisla:: Chisla (int x, int y){   // определение конструктора
  a = x; b = y;                     // инициализация a, b
}
int Chisla::sum(){                  // определение функции sum()
  return a + b;
}
int Chisla::mult(){                 // определение функции mult()
  return a * b;
}
Chisla::show(){                     // определение функции show()
  cout<<"число a = "<<a<<" число b = "<<b<<endl;
}
int main(){                         // функция main()
  system ("cls");
  int n, m;
  cout<<"vvedi chisla: ";
  cin>>n>>m;                        //ввод чисел
  Chisla ob(n, m);                  // создание объекта ob типа Chisla
  ob.show();                        // вызов функции show(), вывод чисел
```

```
  int s, pr;
  s = ob.sum();                    // вызов функции sum()
  pr = ob.mult();                  // вызов функции mult()
  cout<<"summa = "<<s<<" mult = "<<pr<<endl;
//  cout<<"summa = "<<ob.sum()<<" mult = "<<ob.mult()<<endl;
  system("pause");
  return 0;
}
```

// Пример 1.2. Нахождение суммы цифр натурального числа.
// Использование конструктора.

```
#include <iostream.h>
class Cifra{                       // определение класса Cifra
  long chislo;                     // число
  int sum;                         // сумма цифр
public:
  Cifra(long n);                   // конструктор с параметром
  summa();                         // функция получения суммы цифр числа
  show();                          // функция вывода суммы цифр числа
};
Cifra::Cifra(long n){              // определение конструктора
  chislo = n;                      // инициализация переменной chislo
}
Cifra::summa(){                    // определение функции summa()
  sum = 0;
  long rab = chislo;
  while(rab){                      // цикл для выделения цифр числа
    sum = sum + rab % 10;
    rab = rab / 10;
  }
}
Cifra::show(){                     // определение функции show()
  cout<<"summa cifr= "<<sum<<endl;
}
int main(){                        // функция main()
  system ("cls");
  long n;
  cout<<"vvedi chislo: ";
  cin>>n;                          //ввод числа
  Cifra ob(n);                     // создание объекта ob типа Cifra
```

```
  ob.summa();              // вызов функции summa()
  ob.show();               // вызов функции show()
  system("pause");
  return 0;
}
```

Результат:

```
vvedi chislo: 31456
summa cifr = 19
```

```
// Пример 1.3. Нахождение суммы цифр натуральных чисел
// из диапазона от n до m. Использование конструктора.
#include <iostream.h>
class Cifra{
   long n;
   long m;
   int sum;
public:
   Cifra (long x, long y);   // конструктор с параметром
void summa();                // функция получения суммы цифр числа
void show(int a);            // функция вывода суммы цифр числа
};
Cifra::Cifra(long x, long y){      // определение конструктора
   n = x;                          // инициализация n, m
   m = y;
}
void Cifra::summa(){               // определение функции summa()
  for(long i=n;i<m; i++){
   sum=0;
   long rab = i;
   while(rab){                     // цикл по выделению цифр числа
     sum=sum+rab%10;               // и накоплению суммы в переменной
     rab/=10;                      // класса sum
    }
   show(i);
  }
}
void Cifra::show(int i){           // определение функции show()
   cout<<"summa cifr chicla "<<i<<" = "<<sum<<endl;
}
int main(){                        // функция main()
   long n, m;
```

```
    cout<<"vvedi chisla: ";
    cin>>n>>m;                    //ввод числа
    Cifra ob(n, m);               // создание объекта ob типа Cifra
    ob.summa();                   // вызов функции summa()
    system("pause");              // задержка экрана
    return 0;
}
```

// **Пример 1.4.** Определить, образуют ли цифры натурального числа
// возрастающую последовательность?

```
#include <iostream.h>
class UporCifr{
    int a;                        // private-члены
public:
    UporCifr(int n);
    void show();
    void vozr();
};
UporCifr::UporCifr(int n){        // определение конструктора
   a = n;                         // инициализация
 }
void UporCifr::show(){
 cout<<" a= "<<a<<endl;
}
void UporCifr::vozr(){
  int r = a;  int flag = 0;
  int cpr, csl;
  cpr = r%10;  r = r / 10;
  while (r>0){
    csl = r%10;
    if(csl > cpr){flag = 1; break;}  // знак > так как выделение цифр
    r = r / 10;                   // начинается с последней цифры
    cpr = csl;
  }
  if(flag == 0)cout<<"uporaydocheni";
  else cout<<"no uporaydocheni";
}
int main(){                       // функция main()
    int chislo;
    cout<<"vvedi chislo: ";  cin>>chislo;        //ввод числа
```

```
    UporCifr ob(chislo);          // создание объекта ob
    ob.show();                    // вызов функции show()
    ob.vozr();                    // вызов функции vozr()
    system("pause");              // задержка экрана
    return 0;
}
```

// **Пример 1.5.** Определить, можно ли построить треугольник по трём
// сторонам, и если да, то найти площадь треугольника.

```
#include <iostream.h>
#include <math.h>
class Pl{
  float a, b, c;
  float p;    float s;
public:
  Pl(float x,float y,float z){a = x; b = y; c = z;};
  int per();
  sq();
  show();
};
int Pl::per(){
  int flag;
  if((a+b > c) && (b+c > a) && (a+c > b)){
    p=((a+b+c)/2);cout<<"p="<<p<<endl;
    flag=1;
  }
  else{flag=0; cout<<"nelzya"<<endl;}
  return flag;
}
Pl::sq(){s=sqrt(p*(p-a)*(p-b)*(p-c));}
Pl::show(){
  cout<<" a = "<<a<<" b = "<<b<<" c = "<<c<<endl;
  cout<<"square= "<<s<<endl;
}
main(){
  system ("cls");
  int rez;
  float a, b, c;
  cout<<"vvedi chisla: ";
  cin>>a>>b>>c;
```

```
  Pl ob(a,b,c);
  rez = ob.per();
  if(rez!=0){ ob.sq(); ob.show();}
  system("pause");                          // задержка экрана
    return 0;
}
```

Варианты контрольных заданий

1. Найти среднее арифметическое цифр натурального числа ***n***, отличных от заданной цифры ***k***.

2. Определить количество цифр, меньших 5, используемых при записи натурального числа ***n***.

3. Получить все четырехзначные числа, сумма цифр которых равна заданному числу ***n***.

4. Выяснить, образуют ли цифры данного натурального числа ***n*** убывающую последовательность.

5. Найти все четные четырехзначные числа, цифры которых следуют в порядке возрастания или убывания.

6. По заданному натуральному числу ***n*** получить число ***m***, записанное цифрами исходного числа, взятыми в обратном порядке.

7. Получить все четырехзначные числа, в записи которых встречаются только цифры 0, 2, 3, 7.

8. Выяснить, есть ли в записи натурального числа ***n*** две одинаковые цифры.

9. Получить все четырехзначные целые числа, в записи которых нет одинаковых цифр.

10. Определить количество и произведение чётных цифр в натуральном числе ***n***.

11. Определить количество и произведение нечётных цифр в натуральном числе ***n***.

12. Определить сумму цифр и их количество, используемых при записи натурального числа ***n***, которые меньше заданной цифры ***k***.

13. Определить максимальную цифру в натуральном числе ***n*** и подсчитать, сколько раз она в нём встречается.

14. Выяснить, является ли разность максимальной и минимальной цифр заданного натурального числа ***n*** чётной, и выдать соответствующее сообщение.

15. Даны натуральные числа ***n*** и ***m***. Найти сумму ***k*** младших цифр числа ***n*** и ***k*** старших цифр числа ***m***.

16. Вычислить сумму цифр натурального числа ***n***, находящихся на нечётных позициях, и произведение цифр, находящихся на чётных позициях (нумерация позиций идёт слева направо).

17. Определить, является ли число $2n+m$ симметричным, т. е. запись числа содержит чётное количество цифр и совпадают его левая и правая половинки.

18. Поменять местами ***k*** старших разрядов с ***k*** младшими разрядами заданного натурального числа n.

19. По заданному натуральному числу ***n*** получить число ***m***, записанное цифрами исходного числа в обратном порядке.

20. Найти наибольший общий делитель (**НОД**) натуральных чисел ***n*** и ***m***, используя алгоритм Евклида (вычитаем из большего числа меньшее до тех пор, пока они не сравняются).

21. Найти наименьшее общее кратное (**НОК**) натуральных чисел ***n*** и ***m*** (НОК(***n***, ***m***) = Abs(***n * m***) / НОД (***n***, ***m***)).

22. Заданное натуральное число ***n*** разложить на простые множители.

23. Выяснить, образуют ли цифры натурального числа ***n*** возрастающую (убывающую) последовательность.

24. Выяснить, есть ли в записи натурального числа ***n*** последовательность из ***k*** одинаковых цифр.

25. В заданном натуральном числе ***n*** найти наибольшую по длине возрастающую (убывающую) последовательность цифр.

26. Среди последовательности натуральных чисел $n_0\ n_1 \ldots n_m$ найти числа и их сумму, которые равны сумме факториалов своих цифр.

27. Среди натуральных чисел $n_0\ n_1 \ldots n_m$ найти число с максимальной суммой своих простых делителей

28. Преобразовать числа заданной последовательности натуральных чисел $n_0\ n_1 \ldots n_m$ так, чтобы цифры каждого числа образовывали максимально возможные числа.

29. Для каждого числа заданной последовательности натуральных чисел $n_0\ n_1 \ldots n_m$ установить, можно ли вычеркнуть в нем некоторые цифры, чтобы сумма оставшихся равнялась заданному числу ***k***.

30. Найти все четырёхзначные числа из последовательности натуральных чисел $n_0\ n_1 \ldots n_m$ и подсчитать их количество:

a) числа, в записи которых не повторяются цифры;
b) числа, в записи которых цифры упорядочены;
c) числа, в записи которых встречаются только цифры 0, 1, 2, 3;
d) числа, сумма цифр которых равна заданному числу ***k***;
e) числа, кратные 45, две средние цифры которых 7 и 9.

31. Найти все пятизначные натуральные числа из диапазона [***n, m***] и подсчитать их количество:

a) числа, средняя цифра которых равна сумме крайних цифр;
b) числа, в записи которых не повторяются цифры;
c) числа, в записи которых цифры упорядочены по возрастанию;
d) наименьшее число, кратное ***N*** такое, что первая его цифра равна пяти и все цифры различны.

32. Найти числа из диапазона [***n, m***], для которых куб суммы его цифр равен самому числу.

33. Найти сумму чисел из диапазона [***n, m***], в записи которых используются только цифры 3, 5, 7.

34. Найти все числа-палиндромы на отрезке [***n, m***], при возведении которых в квадрат получают также числа-палиндромы. Число называется палиндромом, если его запись читается одинаково слева направо и справа налево, например 12321.

35. Даны натуральные числа ***p*** и ***q***. Найти все натуральные числа, меньшие ***p*** и взаимно простые с ***q***.

36. Найти все простые числа, не превосходяшие заданное натуральное число ***n***, двоичная запись которых представляет собой симметричную последовательность нулей и единиц (начинающуюся единицей).

37. Даны натуральные числа ***p*** и ***q***. Найти все делители числа ***q***, взаимно простые с ***p***.

38. Дано натуральное число ***n*** > 13. Получить все пары простых чисел, разность между которыми равна 4, а сами числа меньше ***n***.

39. Дано натуральное число ***n*** > 19. Получить четвёрки простых чисел, принадлежащих одному десятку.

40. Определить, является ли число ***n*** совершенным. Совершенное число ***n*** равно сумме всех своих делителей, не превосходящих ***n***. Например, 6 = 1 + 2 + 3 или 28 = 1 + 2 + 4 + 7 + 14.

41. Найти все совершенные числа из диапазона натуральных чисел [***n, m***] (на сегодня известно 24 совершенных числа; все они чётные; четвёртое число не превышает значения 9999).

42. Два натуральных числа называют дружественными, если каждое из них равно сумме всех делителей второго не считая самого этого числа. Найти все пары дружественных чисел на отрезке [***n, m***].

43. Заданное натуральное число ***n*** преобразовать так, чтобы его цифры следовали в порядке возрастания (убывания).

44. Для заданного натурального числа ***n*** найти все меньшие его автоморфные числа. Автоморфным называется число, совпадающее с младшими цифрами своего квадрата, например, $5^2 = 25$, $6^2 = 36$, $25^2 = 625$.

45. Определить, сколько раз каждая десятичная цифра встречается в записи натурального числа.

ЗАДАНИЕ 2. ОДНОМЕРНЫЕ МАССИВЫ – ЧЛЕНЫ КЛАССА

Цель задания – получение практических навыков создания и использования классов в случае, когда членами класса являются одномерные массивы.

Основные понятия

Одномерный массив представляет собой группу однотипных элементов, имеющих общее имя, при этом доступ к каждому элементу массива осуществляется по имени массива и индексу элемента. Индекс первого элемента всегда **0,** далее следуют **1**, **2**, и т.д. Элементы массива занимают непрерывный участок памяти компьютера и располагаются последовательно друг за другом. Обращение к элементу массива, индекс которого не является допустимым, приводит к возникновению ошибок, вызванных обращением к области памяти, не принадлежащей к массиву.

Количество элементов массива называют **размерностью массива.**

Перед использованием в программе массив следует объявить, т.е. сообщить компилятору, что будет использоваться массив с данным именем, типом и количеством элементов:

```
int a[10];                  // объявление массива a из 10 целых чисел
int b[5] = {1, 2 ,3, 4, 5}; // объявление и иинициализация массива b
int c = b[2];               // c = 3
```

Имя массива является константным указателем, его нельзя изменять, но можно присваивать указателям соответствующего типа:

```
int  a[4];
int  *pa;              // указатель на тип int
pa = a;                // указателю pa присваивается адрес массива a
for(int i = 0; i < 4; i++)
  cout<<a[i]<<' ';     // обращение к элементам массива
// или cout<<*(a+i); или cout<<pa[i]; или cout<<*(pa+i);
```

Отметим, что прибавление к указателю единицы обеспечивает переход к следующему элементу массива.

Массивы с фиксированной размерностью называют **статическими**. Однако часто при решении задач размеры массива заранее не известны и определяются в ходе решения поставленной задачи. Такие массивы называют **динамическими.**

Динамический одномерный массив можно создать с помощью *операции* ***new***, который выделяет в динамической области памяти участок для размещения массива соответствующего типа. Освобождают память *операцией* ***delete***:

```
int n;
cin>>n;              // вводится размерность массива n
int *mas;            // указатель на целый тип
mas = new int[n];    // выделение памяти под массив
delete [] mas;       // удаление памяти
```

Если с помощью оператора ***new*** невозможно выделить требуемый объём памяти, то оператор ***new*** возвратит нулевой указатель.

Явно инициализировать динамические массивы нельзя.

Постановка задания

Определить класс для решения задач, использующих одномерные массивы, предусмотрев в нём функции для инициализации, ввода, вывода переменных, массивов, объявленных в классе, функции для доступа к членам класса и для работы с объектами данного класса. Написать программу, которая демонстрирует использование созданного класса для работы с одномерными массивами (статическими, динамическими).

Примеры выполнения задания

// Пример 2.1. Определение класса для работы с одномерным
// статическим массивом. Нахождение суммы элементов массива.

```
#include <iostream.h>
#include<stdlib.h>
const int max_n = 50;
class Massiv{        // определение класса Massiv
  int a[max_n];      // объявление массива a
  int n;             // размерность массива
  int sum;           // сумма элементов массива
public:
  Massiv(int k);     // конструктор с параметром
  summa();           // функция нахождение суммы элементов массива
  show();            // функция вывода массива
};
```

```
Massiv::Massiv(int k){          // определение конструктора
  n = k;                        // инициализация n
  for(int i = 0; i < n; i++)    // инициализация массива
    a[i] = random(n);           // случайными числами
}
Massiv::summa(){                // определение функции summa()
  sum = 0;
  for(int i = 0; i < n; i++)
    sum = sum + a[i];
}
Massiv::show(){                 // определение функции show()
  cout<<"Massiv a: ";
  for(int i = 0; i < n; i++){
    cout.width(4);
    cout<<a[i]<<" ";
   }
  cout<<endl;
  cout<<"Summa elementov massiva a= "<<sum<<endl;
}
int main(){
  int kol;
  cout<<"Vvedi chislo elementov:";  cin>>kol;
  Massiv ob(kol);           // создание объекта ob типа Massiv
  ob.summa();               // вызов функции summa()
  ob.show();                // вызов функции show()
  system("pause");          // задержка экрана
  return 0;
}
```

Результат:

```
Vvedi chislo elementov:6
massiv a: 2 4 0 2 0 0
Summa elementov massiva a = 8
```

// Пример 2.2. Определение класса для работы с одномерным
// динамическим массивом. Нахождение суммы элементов массива.
// Использование деструктора.

```
#include <iostream.h>
#include <stdlib.h>
class Massiv{               // определение класса Massiv
  int n;
  int *p;                   // указатель на тип int
```

```
  int sum;
public:
  Massiv(int k);              // конструктор
  ~Massiv(){delete [] p;}     // деструктор определён в классе
  summa();
  show();
};

Massiv::Massiv(int k){        // определение конструктора
  n = k;                      // инициализация n
  p = new int[n];             // выделение динамической памяти под массив
  if(p == NULL){
    cout<<"no memory"<<endl;
    exit(1);
  }
  for(int i = 0; i < n; i++)      // инициализация массива
    p[i] = random(n);
}

Massiv::summa(){              // определение функции summa()
  sum = 0;
  for(int i = 0; i < n; i++)
    sum = sum + p[i];
}

Massiv::show(){               // определение функции show()
  cout<<"Massiv=="<<endl;
  for(int i = 0; i < n; i++){
   cout.width(4);
   cout<<p[i]<<" ";
  }
  cout<<"\n Summa elementov massiva = "<<sum<<endl;
}
int main(){
  int kol;
  cout<<"Vvedi chislo elementov:";     cin>>kol;
  Massiv ob(kol);             // создание объекта ob типа Massiv
  ob.summa();
  ob.show();
 system("pause");             // задержка экрана
  return 0;
}
```

Результат:

```
Vvedi chislo elementov:7
Massiv==
3 2 1 4 4 0 1 6
Summa elementov massiva = 21
```

Варианты контрольных заданий

1. В массиве A(N) определить количество пар одинаковых соседних элементов и количество перемен знаков.
2. В массиве A(N) поменять местами первый положительный и последний отрицательный элементы, а также найти сумму элементов, стоящих между ними.
3. В массиве A(N) найти максимальное и минимальное отклонения значений элементов от их среднего значения.
4. В массиве A(N) найти номер элемента, сумма которого со следующим за ним элементом максимальна, и номер элемента, сумма которого с предыдущим элементом минимальна.
5. В массиве A(N) найти минимальный из положительных элементов и максимальный из отрицательных элементов.
6. В массиве A(N) найти первый отрицательный элемент, предшествующий максимальному элементу, и последний положительный элемент, стоящий за минимальным элементом.
7. В массиве A(N) найти сумму элементов, расположенных между первым и вторым нулевыми элементами.
8. В массиве A(N) поменять местами последний отрицательный элемент с максимальным элементом.
9. В массиве A(N) найти максимальный элемент среди четных элементов и минимальный элемент среди нечетных элеметов.
10. В массиве A(N) все четные элементы заменить максимальным элементом, а нечетные – минимальным элементом.
11. В массиве A(N) найти первый отрицательный элемент, кратный заданному числу ***p***, заменить его индексом и поставить в начало массива.
12. В массиве A(N) выбрать все элементы, встречающиеся один раз.
13. В массиве A(N) выбрать без повторений все элементы, встречающиеся более одного раза.
14. В массиве A(N) подсчитать количество различных элементов.
15. В массиве A(N) найти максимальный из элементов, встречающихся в массиве только по одному разу.
16. В массиве A(N) найти наименьшее количество элементов, которые надо удалить из данного массива, чтобы осталась возрастающая последовательность.

17. Из всех участков массива A(N), сплошь состоящих из нулей, выбрать самый длинный и отметить индексы его начала и конца.

18. В массиве A(N) найти наибольшую по длине возрастающую последовательность чисел и определить её длину.

19. В массиве A(N) найти последовательность наибольшей длины из знакочередующихся элементов и отметить индексы её начала и конца.

20. В массиве A(N) найти наименьшую по длине убывающую последовательность чисел и отметить индексы её начала и конца.

21. Преобразовать заданный массив A(N) натуральных чисел так, чтобы цифры каждого его элемента были записаны в обратном порядке.

22. В массиве A(N) переставить элементы, стоящие на нечётных местах с элементами, стоящими на чётных местах.

23. Удалить из массива A(N) элементы, стоящие за первым максимальным элементом, количество цифр которых равно заданному ***k***.

24. Из массива A(N) удалить все отрицательные элементы, стоящие перед первым наименьшим элементом.

25. Удалить из массива A(N) элементы, стоящие между первым максимальным элементом и последним отрицательным элементом.

26. Из массива A(N) удалить все четные положительные элементы, стоящие после первого максимального элемента.

27. Из массива A(N) удалить все элементы, стоящие между первым минимальным и последним максимальным элементами.

28. Удалить из массива A(N) последнюю группу элементов, представляющих собой знакочередующийся ряд.

29. Преобразовать заданный массив A(N), оставив из всех цепочек идущих подряд одинаковых элементов только один такой элемент.

30. В массив A(N) вставить после первого максимального элемента ***k*** наименьших элементов.

31. В массив A(N) вставить максимальный элемент после каждого четного отрицательного элемента.

32. В массив A(N), упорядоченный по возрастанию, вставить ***k*** различных чисел, не нарушая упорядоченности.

33. Для каждого из элементов массива A(N) определить число его вхождений в массив.

34. Элемент называется локальным минимумом (максимумом), если у него нет соседа, меньшего (большего), чем он сам. Найти все локальные минимумы и максимумы в заданном массиве A(N).

35. Элементы массива A(N) упорядочить в порядке возрастания или убывания в зависимости от признака.

36. Получить из двух упорядоченных по возрастанию массивов третий массив, также упорядоченный по возрастанию.

37. В массиве A(N) встречаются лишь числа от 1 до 20. Упорядочить элементы массива в порядке частоты встречаемости чисел.

38. Если максимальный элемент массива A(N) число четное, то все элементы, стоящие за ним , расположить в порядке возрастания.

39. Все положительные элементы массива A(N), значения которых находятся в заданных пределах, примкнуть к последнему максимальному элементу, расположив по убыванию.

40. Положительные элементы массива A(N) переставить в конец массива и расположить в порядке убывания.

41. Отрицательные элементы массива A(N) переставить в начало массива и расположить в порядке возрастания.

42. В целочисленном массиве A(N) элементы массива, являющиеся простыми числами, расположить в порядке возрастания.

43. В массиве A(N) найти последовательность максимальной длины из элементов, которые имеют чередующиеся знаки и расположены по возрастанию модулей.

44. Умножить два целых больших числа. Для таких чисел вещественный тип неприемлем, а типа ***long int*** недостаточно, так как в числе больше 10 цифр. Для хранения такого числа предлагается использовать одномерный массив, каждый элемент которого представляет собой одну десятичную цифру числа.

ЗАДАНИЕ 3. ДВУМЕРНЫЕ МАССИВЫ – ЧЛЕНЫ КЛАССА

Цель задания – получение практических навыков создания и использования классов в случае, когда членами класса являются двумерные массивы.

Основные понятия

Двумерный массив – матрица – представляет собой массив одномерных массивов. Каждый элемент матрицы имеет два индекса: номер строки и номер столбца, которые указываются в отдельных квадратных скобках. Например:

```
int a[3][4];                  // первый индекс изменяется от 0 до 2,
                              // второй индекс – от 0 до 3
int b[2][2] = {{1, 2},        // явная инициализация, строка 0
               {3, 4}};       // явная инициализация, строка 1
```

Под обычный двумерный массив (**статический массив**) при объявлении выделяется сплошной участок памяти. Массив размещается в выделенной памяти по строкам. Для создания, просмотра, обработки таких массивов используются вложенные циклы.

При создании **двумерного динамического массив**а сначала выделяется память под одномерный массив указателей на адреса одномерных массивов (строк). Затем выделяется память под каждый из одномерных массивов, при этом их адреса заносятся в соответствующий элемент массива указателей. Например, при создании динамического массива для работы с матрицей из ***n*** строк и ***m*** столбцов, следует выполнить следующие действия:

```
int n, m;
int **matr;                 // указатель на массив указателей
cin>>n; cin>>m;
matr = new *int[n];         // память под массив указателей на строки
for(int i; i < n; i++)
  matr[i] = new int[m];     // память для каждой строки
```

При создании двумерных динамических массивов следует помнить, что выделенная динамическая память для него не представляет собой сплошной участок, поскольку она выделяется несколькими операторами ***new.***

Удаление двумерного массива происходит в несколько этапов:

```
for(int i = 0; i < n; i++)     // удаление одномерных массивов (строк)
  delete [] matr[i];
delete [] matr;                // удаление массива указателей
```

Для ссылок к элементам динамического массива с индексами ***i*** и ***j*** допускаются выражения:

```
matr[i][j];
или    *(matr[i] + j);
или   *(*( matr + i) + j ).
```

Постановка задания

Определить класс для решения задач, использующих двумерные массивы, предусмотрев в нём функции для инициализации, ввода, вывода переменных, массивов, объявленных в классе, функции для доступа к членам класса и для работы с объектами данного класса. Написать программу, которая с помощью меню демонстрирует использование созданного класса для работы с двумерными массивами (статическими, динамическими) и осуществляет проверку всех методов класса.

Примеры выполнения задания

// **Пример 3.1.** Определение класса для работы с двумерным статическим
// массивом. Нахождение максимального элемента в каждой строке.

```
#include <iostream.h>
#include <stdlib.h>
const int max_n = 10;
class Matr{                  // определение класса Matr
  int a[max_n][max_n];       // матрица a
  int b[max_n];              // одномерный массив b
  int n;                     // число строк в матрице a
  int m;                     // число столбцов в матрице a
public:
  Matr(int k, int l);        // конструктор
  max_strok();               // функция поиска максимального элемента
  show_matr();               // функция вывода матрицы
  show_mas();                // функция вывода массива
};
Matr::Matr(int k, int l){    // определение конструктора
  n = k;  m = l;
  for (int i = 0;i < n; i++)   // инициализация матрицы
    for (int j = 0; j < m; j++)
      a[i][j] = random(n*m);
}
Matr::max_strok(){           // определение функции max_strok()
  int max;
  for(int i = 0; i < n; i++){
    max = a[i][0];
    for (int j = 0; j < m; j++)
     if( a[i][j] > max) max = a[i][j];
    b[i] = max;              // максимальный элемент строки –> b[i]
  }
}
Matr::show_matr(){           // определение функции show_matr()
  cout<<"Matrix== "<<endl;
  for (int i = 0; i < n; i++){
    for (int j = 0; j < m; j++){
      cout.width(4);   cout<<a[i][j]<<" ";
    }
```

```
    cout<<endl;
   }
  }
  Matr::show_mas(){          // определение функции show_mas()
   cout<<"Massiv b== "<<endl;
   for (int i = 0; i < n; i++)
      cout<<b[i]<<" ";
   cout<<endl;
  }
  int main(){                // функция main()
   int n, m;
   cout << "Vvedi kolichestvo strok i stolbcov ";  cin>>n>>m;
   Matr ob(n, m);            // создание объекта ob типа Matr
   ob.show_matr();
   ob.max_strok();    ob.show_mas();
   system("pause");          // задержка экрана
   return 0;
  }
```

Результат:

```
Vvedi kolichestvo strok i stolbcov: 3 4
Matrix==                   Massiv b==
  2   1   0   2               2   0   2
  0   0   0   0
  0   2   2   0
```

// **Пример 3.2.** Определение класса для работы с динамическим
// двумерным массивом. Использование деструктора.

```
  #include <iostream.h>
  #include <stdlib.h>
  class Matr{
     int **a;                // указатель на массив указателей
     int n, m;               // размерности матрицы
  public:
     Matr(int k, int l);     // конструктор
     ~Matr();                // деструктор
     void show_matr();       // функция вывода матрицы
  };
  Matr::Matr(int k, int l){          // определение конструктора
   n = k;  m = l;
   a = new int*[n];                  // память под массив указателей
```

```
  if(a == NULL){
   cout<<"No memory"<<endl;
   exit(1);
  }
  for(int i = 0;i < n; i++){
    a[i] = new int[m];                  // выделение памяти для строки
    if(a[i] == NULL){
      cout<<"No memory"<<endl;
      exit(1);
    }
    for(int j = 0; j < m; j++)          // заполнение i-ой строки
      *(a[i]+j) = random(100);          // или a[i][j] = random(100);
  }
}
Matr::~Matr(){                          // определение деструктора
  for(int i = 0; i < n; i++)
    delete [] a[i];                     // удаление строк
  delete [] a;                          // удаление массива указателей
}
void Matr::show_matr(){                 // определение show_matr()
  cout<<"Matrix== "<<endl;
  for (int i = 0; i < n; i++){
   for (int j = 0; j < m; j++){
    cout.width(4);
    cout<<a[i][j]<<" ";
   }
  cout<<endl;
  }
}
int main(){
  int n, m;
  cout<<"Vvedi n, m: ";  cin>>n>>m;
  Matr ob(n, m);                // создание объекта ob типа Matr
  ob.show_matr();               // вызов функции show_matr()
   system("pause");             // задержка экрана
  return 0;
}
```

Результат:

```
Vvedi n, m: 3 4
```

Matrix==
40 20 12 76
88 72 92 12
13 60 77 19

Варианты контрольных заданий

1. В массиве A(N, M) найти:
   a) среднее арифметическое элементов по периметру;
   b) номер строки с максимальным числом нулей;
   c) максимальный элемент строк, не содержащих нулей;
   d) сумму элементов столбцов, содержащих хотя бы один ноль.
2. В массиве A(N, M) найти количество строк, элементы которых образуют арифметическую прогрессию.
3. Строки массива A(N, M), не содержащие положительных элементов, заменить нулевыми элементами.
4. В массиве A(N, M) подсчитать количество строк и столбцов, составленных из попарно различных чисел.
5. В массиве A(N, M) найти минимум среди сумм элементов диагоналей, параллельных главной диагонали и максимум среди сумм элементов диагоналей, параллельных побочной диагонали.
6. В массиве A(N, M) найти строку, среднее арифметическое положительных элементов которой является наименьшим.
7. В массиве A(N, M) найти сумму тех элементов, в записи которых используются только цифры 0, 1, 2, 3.
8. В массиве A(N, M) найти сумму элементов тех столбцов, все элементы которых кратны заданному числу ***p***.
9. В матрице A(N, N) найти первую строку, не содержащую отрицательных элементов, и умножить ее как вектор на матрицу A.
10. В массиве A(N, M) найти строку и столбец, для которых количество перемен знаков максимально.
11. В массиве A(N, N) найти максимальную сумму среди сумм элементов диагоналей, параллельных главной диагонали.
12. В массиве A(N, N) поменять местами максимальные элементы нижнего и верхнего треугольников относительно главной диагонали.
13. Минимальные элементы строк массива A(N, M), не содержащих отрицательных элементов, заменить произведением их индексов.
14. В массиве A(N, M) найти максимальный и минимальный элементы среди элементов тех строк, которые упорядочены по возрастанию или по убыванию.

15. В массиве A(N, M) заменить на обратные величины максимальные элементы строк, не содержащих четных элементов.

16. В массиве A(N, M) сменить знаки элементов строки и столбца, на пересечении которых находится минимальный элемент.

17. В массиве A(N, M) минимальный элемент каждого столбца заменить суммой положительных элементов этого же столбца.

18. Элементы столбцов целочисленного массива A(N, M), не содержащих положительных элементов, заменить суммой их цифр.

19. В каждой строке массива A(N, M) сменить знаки элементов, находящихся между первым и последним максимальными элементами данной строки.

20. В массиве A(N, M) сменить знаки минимальных элементов столбцов, не содержащих отрицательных элементов.

21. В массиве A(N, M) поменять местами столбцы, содержащие максимальное и минимальное количество чётных элементов.

22. В каждой строке массива A(N, M) найти и удалить максимальные и минимальные элементы.

23. В массиве A(N, M) удалить столбец с минимальным произведением элементов и строку с максимальной суммой элементов.

24. В массиве A(N, M) расположить столбцы в порядке убывания их минимальных элементов.

25. массиве A(N, M) упорядочить элементы каждой строки верхнего треугольника по возрастанию, нижнего по убыванию, диагональные элементы оставить без изменения.

26. В массиве A(N, M) расположить строки так, чтобы сначала шли строки, у которых положительных элементов больше, чем отрицательных, затем – с одинаковым числом положительных и отрицательных элементов и последними, чтобы шли строки, имеющие положительных элементов меньше, чем отрицательных.

27. В массиве A(N, M) часть строк состоит из нулей. Переместить нулевые строки в конец массива.

28. В массиве A(N, M) расположить строки в порядке возрастания элементов главной диагонали (побочной диагонали).

29. В массиве A(N, M) элементы строк, начинающихся с отрицательного элемента, расположить в порядке возрастания.

30. В массиве A(N, M) элементы столбцов, не содержащих нулевых элементов, расположить в порядке убывания.

31. В массиве A(N, M) расположить строки в порядке возрастания их максимальных элементов.

32. В массиве A(N, M) расположить столбцы в порядке возрастания количества перемен знаков в каждом столбце.

33. В массиве A(N, M) строки с нечётными индексами упорядочить по убыванию, а с чётными – по возрастанию.

34. В массиве A(N, M) строку с максимальным количеством знакочередующихся элементов поменять местами с первой строкой.

35. массиве A(N, N), где N – четное, переставить друг с другом центрально-симметричные квадраты.

36. В массиве A(N, M) переставить строки в порядке убывания количества содержащихся в них положительных элементов.

37. В каждой строке массива A(N, M) найти максимальный из элементов, встречающихся в строке только один раз.

38. Симметричную матрицу A(N, N), заданную верхним треугольником в виде одномерного массива, умножить на вектор B(N).

ЗАДАНИЕ 4. МАССИВЫ ОБЪЕКТОВ. УКАЗАТЕЛИ

Цель задания – изучение приёмов объектного программирования для работы с массивами объектов, указателями на объекты и массивы объектов.

Основные понятия

Массив объектов создаётся аналогично массиву переменных стандартного типа:

```
myClass  a[4];        // одномерный массив объектов a типа myClass
myClass  b[2][4];     // двумерный массив объектов b типа myClass
```

Если класс содержит конструктор, массив объектов может быть проинициализирован, причём конструктор будет вызван столько раз, сколько элементов содержит массив:

```
myClass ob[4] = {1, -2, 5, 6};
```

Известно, что доступ к членам объекта осуществляется с помощью *операции точка* (.). Однако доступ к члену объекта можно получить и через указатель на этот объект. В этом случае используется *операция стрелка* **(->)**:

```
myClass ob;          // объект типа myClass
myClass *uk;         // указатель на тип myClass
uk = &ob;            // указателю присваивается адрес объекта ob
ob.set(10);          // вызов функции через оператор “точка”
uk–>set(10);         // вызов функции через оператор “стрелка”
```

При работе с массивом объектов также можно использовать указатели:

```
myClass ob[5];      // массив объектов типа myClass
myClass *p;         // указатель на тип myClass
p = ob;             // указатель на нулевой элемент массива объектов ob
p++;                // указатель на следующий элемент массива
p--;                // указатель на предыдущий элемент массива
```

C++ содержит специальный указатель ***this***, который автоматически передаётся любой функции-члену класса при её вызове и указывает на объект, генерирующей вызов. При этом ****this*** представляет сам объект. Дружественным функциям указатель ***this*** не передаётся.

Динамический объект можно создать с помощью указателя и *операции* ***new:***

```
Samp*uk = new Samp(4, 6);
```

При размещении объекта автоматически вызывается его конструктор, и объекту передаются значения 4 и 6.

Для разрушения динамического объекта используется *операция* ***delete***, который может помещаться в деструкторе:

```
delete uk;
```

Динамический массив объектов создаётся аналогично обычному динамическому массиву:

```
Samp *p;            // указатель на тип Samp
p = new Samp[4];    // одномерный динамический массив объектов
```

Конструктор будет вызван четыре раза, если он присутствует в классе.

Динамический массив объектов нельзя явно инициализировать, поэтому наряду с другими конструкторами в классе должен присутствовать конструктор по умолчанию.

Для освобождения памяти, занимаемой динамическим массивом объектов, используется оператор ***delete.*** Например, для удаления динамически размещённого одномерного массива объектов следует использовать следующую форму оператора ***delete***:

```
delete [ ] p;
```

Постановка задания

Определить класс, предусмотрев в нём функции для инициализации, ввода, вывода переменных членов класса, функции для доступа к членам класса. Сформировать динамический массив объектов данного класса и определить функции для работы с созданным массивом. Написать программу, которая с помощью меню демонстрирует использование функций созданного класса.

Примеры выполнения задания

// Пример 4.1. Создается класс **Student**. Формируется динамический массив
// объектов данного класса. При тестировании программы на экран выводится:
// сформированный список студентов; список студентов заданного факультета;
// список студентов для заданных факультета и курса.

```
#include <string.h>
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>
struct date{                          // структура – дата рождения
  int mon;
  int year;
};
const size=10;
class Student {                       // класс Student
   char fam[size];
   date t;
   char fac[size];
   int kurs;
public:
   Student();
   char* getfac();
   int   getkurs();
   void  show();
};
Student::Student(){
      cout << "Input fam\n";      cin >> fam;
      cout << "Input date of birthday\n";
      cout << "Month:" << endl;   cin >> t.mon;
      cout << "Year:" << endl;    cin >> t.year;
      cout << "Input facultet\n"; cin >> fac;
      cout << "Input kurs\n";     cin >> kurs;
}
//=======    show    =================
void Student::show(){
 cout<<"| "<<setw(9)<<fam<<"  | "<<setw(4)<<t.mon<<"    |  ";
 cout<<setw(5)<<t.year<<" | "<<setw(4)<<fac<<"  | "<<setw(2);
 cout<<kurs<<" |"<<endl;
}
```

```
//=======    getfac    ================
char* Student::getfac(){
     return fac;
}
//=======    getkurs    ================
int Student::getkurs(){
     return kurs;
}
void vivod(Student *spis, int n);          // вывод списка студентов
void spisfac(Student *spis,int n);         //вывод списка студентов
                                           // заданного факультета
void spisfackurs(Student *spis,int n);     // список студ. факультета и курса
void shapka();
int main(){
  Student *spis;
  int n, q;
  cout<<"Input a number of students: "<< endl; cin>>n;
  spis=new Student[n];                     //динамический массив объектов
  while(true){
    cout<<"viberite nomer comandi(1-4)"<<endl;
    cout<<"1-write display "<<endl;
    cout<<"2-spisok  fakulteta "<<endl;
    cout<<"3-spisok  fakulteta i kursa"<<endl;
    cout<<"4-vixod"<<endl;    cin>>q;
    switch(q){
      case 1:vivod  (spis, n);break;
      case 2:spisfac(spis, n);break;
      case 3:spisfackurs(spis, n);break;
      case 4:exit(1);
    }
  }
     cout << "Press any key !" << endl;
     delete [] spis;
     system("pause"); return 0;
}
void shapka(){                  //=====вывод шапки=====
 cout<<"|-----------------------------------------|"<<endl;
 cout<<"|  Family  | month | year | fac |kurs|"<<endl;
 cout<<"|-----------------------------------------|"<<endl;
}
```

```
//=====список всех студентов==========================
void vivod(Student *spis, int n){
   shapka();
   for(int i = 0; i < n; i++)
    spis[i].show();
}
//=======список студентов заданного факультета========
void spisfac(Student *spis,int n){
      int i;
      char fac[size];
      cout<<"vivod studentov zadannogo faculty"<<endl;
      cout << "Input faculty" << endl;
      cin >> fac;
      shapka();
      for( i = 0; i < n; i++)
       if(strcmp(spis[i].getfac(),fac)==0)spis[i].show();
}
//=======список студентов заданных факультета и курса==
void spisfackurs(Student *spis,int n){
     int i,k;
     char fac[size];
      cout<<"vivod studentov zadannogo faculty i course"<<endl;
      cout << "Input faculty" << endl;      cin >> fac;
      cout << "Input the course" << endl;
      cin >> k;
      shapka();
      for( i = 0; i < n; i++)
        if ((strcmp(spis[i].getfac(),fac)==0)&&(spis[i].getkurs()==k))
            spis[i].show();
}

void spisfac(Stud spis[], int n){            // функция spisfac()
  int i;
  char fac[20];
  cout<<"vivod studentov zadannogo faculty"<<endl;
  cout<<"Input faculty"<<endl;    cin >>fac;
  for( i = 0; i < n; i++)
    if(strcmp(spis[i].getfac(),fac) == 0)spis[i].show();
}
```

```
void spisfackurs(Stud spis[], int n){      // функция spisfackurs()
  int i, k;
  char fac[20];
  cout<<"vivod studentov zadannogo faculty i course"<<endl;
  cout<<"Input faculty"<<endl;
  cin>>fac;
  cout<<"Input the course"<<endl;     cin>>k;
  for( i = 0; i < n; i++)
    if((strcmp(spis[i].getfac(), fac) == 0) && (spis[i].getkurs() == k))
      spis[i].show();
}
```

Варианты контрольных заданий

1. **Abiturient:** Фамилия, Имя, Отчество, Адрес, Оценки. Вывести:
   a) список абитуриентов, имеющих неудовлетворительные оценки;
   b) список абитуриентов, средний балл у которых выше заданного;
   c) список абитуриентов, средний балл у которых ниже заданного;
   d) список абитуриентов в алфавитном порядке.

2. **Aeroflot:** Номер рейса, Пункт назначения, Время вылета, Информация о наличии свободных мест. Вывести:
   a) информацию об указанном рейсе;
   b) список рейсов и время вылета для заданного пункта назначения;
   c) билет для пассажира, если есть свободные места на указанный рейс, или предложить другой рейс, если на этот рейс нет билетов.

3. **Automobile**: Марка автомобиля, Год выпуска, Номер, Фамилия владельца. Вывести:
   a) список автомашин не старше заданного года выпуска;
   b) список автомашин заданной марки, упорядочив его по году выпуска (фамилии владельца).
   c) сведения об автомашинах в алфавитном порядке.

4. **Boots**: Артикул, Наименование, Размер, Количество пар, Стоимость одной пары. Артикул начинается с буквы D для дамской, с M – для мужской, с C – для детской обуви. Вывести:
   a) информацию о наличии и стоимости обуви артикула XX;
   b) списки дамской, мужской и детской обуви заданного размера, имеющиеся в наличии, и число пар каждой модели;
   c) список наименований всей обуви в алфавитном порядке.

5. **Book**: Автор, Название, Издательство, Год. Вывести:
   a) список книг заданного автора в алфавитном порядке;
   b) список книг, выпущенных заданным издательством;
   c) список книг, выпущенных после заданного года.

6. **Bus**: Фамилия водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации. Вывести:
   a) список автобусов для заданного номера маршрута;
   b) список автобусов, которые эксплуатируются больше 10 лет;
   c) список водителей в алфавитном порядке.

7. **Bus**: Номер автобуса, Фамилия водителя, Возраст водителя, Номер маршрута.Вывести:
   a) списки автобусов, находящихся в парке, и автобусов, находящихся на маршруте, упорядочив их в порядке возрастания номеров автобусов (номеров маршрутов);
   b) средний возраст водителей автобусного парка и фамилии самого старшего и самого младшего водителей;
   c) список водителей в алфавитном порядке и их возраст.

8. **Baggage**: Фамилия пассажира, Количество мест багажа и Вес каждого места багажа. Вывести:
   a) список пассажиров в алфавитном порядке;
   b) фамилию пассажира с наибольшим количеством мест багажа;
   c) фамилию пассажира с наименьшим весом багажа.

9. **Bank**: Номер филиала сбербанка, Номер лицевого счёта вкладчика, Фамилия вкладчика, Размер вклада. Вывести:
   a) для каждого филиала списки вкладчиков, упорядоченные по алфавиту и по сумме вклада
   b) для каждого филиала справку об общей сумме вкладов и средний размер вклада.

10. **Customer**: Фамилия, Имя, Отчество, Адрес, Телефон, Номер кредитной карточки. Вывести:
   a) список покупателей в алфавитном порядке;
   b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

11. **Contest**: Название страны, Вид спорта, Фамилия и Имя призёра, Тип медали (золото, серебро, бронза). Вывести:
   a) список призёров страны **NNN** в алфавитном порядке;
   b) список призёров, упорядоченный по названию страны;
   c) списки призёров мужчин и женщин по типу медали.

12. **Championship**: Название команды, Количество забитых мячей, Количество пропущенных мячей, Количество выигрышей. Вывести:

a) список команд в порядке убывания забитых мячей;
b) список команд в порядке убывания количества выигрышей;

13. **File**: Имя файла, Тип файла, Дата создания, Количество обращений к файлу. Вывести:

a) список файлов (имя, тип), созданных ранее указанной даты;
b) список файлов, упорядоченный по имени (типу, дате создания, количеству обращений);
c) список файлов заданного типа по возрастанию количества обращений к ним.

14. **House**: Фамилия владельца, Адрес, Этаж, Количество комнат, Площадь. Вывести:

a) список квартир, имеющих заданное число комнат;
b) список квартир, отсортированный по размеру площади;
c) список жильцов в алфавитном порядке;
d) список жильцов, имеющих площадь, превосходящую заданную.

15. **Library**: Шифр темы, Шифр книги, Фамилия автора, Название книги, Год издания. Вывести:

a) список книг по заданной теме, выпущенных после заданного года издания;
b) список книг заданного автора в порядке возрастания года издания;
c) список книг для заданного года издания, расположив в алфавитном порядке фамилии авторов.

16. **Phone**: Фамилия, Адрес, Номер, Время междугородних разговоров. Вывести:

a) сведения об абонентах, пользовавшихся междугородней связью;
b) список абонентов в алфавитном порядке.

17. **Phone**: Фамилия абонента, Адрес, Номер телефона. Вывести:

a) список абонентов в порядке возрастания номеров телефонов;
b) список жильцов, фамилии которых начинаются на заданную букву, имеют телефон в доме, адрес которого вводится;
c) список телефонов, начинающихся с цифры X.

18. **Person**: Фамилия, Адрес, Образование, Год рождения. Вывести:

a) список граждан, возраст которых превышает заданный;
b) список граждан с высшим образованием;
c) список граждан в алфавитном порядке.

19. **Product**: Наименование, Цена, Срок хранения. Вывести:
   a) список товаров для заданного наименования;
   b) список товаров для заданного наименования, цена которых не превосходит заданную;
   c) список товаров, срок хранения которых больше заданного.

20. **Student**: Фамилия, Имя, Отчество, Адрес, Факультет, Курс. Вывести:
   a) список студентов заданного факультета;
   b) список студентов заданного курса;
   c) списки студентов для заданных факультета и курса.

21. **Student**: Фамилия и Имя студента, Курс, Группа, Оценки, полученные в сессию. Вывести:
   a) для каждого курса список отличников в алфавитном порядке
   b) для заданного курса список студентов в порядке убывания среднего балла.
   c) для заданных курса список студентов, имеющих неудовлетворительные оценки;
   d) список студентов, сдавших все дисциплины.

22. **Student**: Фамилия и Имя студента, Курс, Группа, Место проживания. Вывести:
   a) список студентов в алфавитном порядке, проживающих в общежитии;
   b) список студентов по курсам, нуждающихся в общежитии;
   c) для каждого курса и группы список и количество студентов, проживающих в своей квартире, в общежитии и на квартире.

23. **Student**: Фамилия и Имя студента, Курс, Группа, Оценки, полученные в сессию. Вывести:
   a) список студентов заданного курса и группы, имеющих неудовлетворительные оценки;
   b) список студентов заданного курса, средний балл которых больше четырёх, упорядочив их по алфавиту.

24. **Swimming**: Номер заплыва, Дистанция, Фамилия участника и Время. Вывести:
   a) список участников всех заплывов в алфавитном порядке;
   b) список участников каждого заплыва, упорядоченный в порядке убывания результатов заплыва;
   c) среднее время во всех заплывах и в каждом заплыве определить трёх победителей.

25. **Train**: Пункт назначения, Номер поезда, Время отправления, Число общих мест, Купейных, Плацкартных. Вывести:

a) список поездов, следующих до заданного пункта назначения, упорядоченный по номеру поезда;
b) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;
c) список поездов, следующих до заданного пункта назначения и имеющих общие места (купейные, плацкартные).

26. **Worker**: Шифр цеха, Фамилия рабочего, Пол, Год рождения, Образование, Год поступления на работу. Вывести:

a) список работников моложе 35 лет, не имеющих среднего образования;
b) для каждого цеха средний возраст мужчин и женщин;
c) для каждого цеха список работников со стажем более 5 лет в алфавитном порядке.

27. **Worker**: Фамилия, Должность, Стаж работ, Зарплата. Вывести:

a) список работников, стаж работы которых превосходит заданное число лет;
b) список работников, зарплата которых превосходит заданную;
c) список работников, занимающих заданную должность.

ЗАДАНИЕ 5. ПЕРЕГРУЗКА ОПЕРАЦИЙ

Цель задания – изучение приёмов объектного программирования с использованием операторных функций, обеспечивающих перегрузку стандартных операторов для работы с объектами данного класса.

Основные понятия

В **C++** операторы рассматриваются как функции, поэтому операции, также как и функции, можно перегружать. Перегружаются почти все операции, за исключением некоторых: **.,** $\mathbf{.^*}$**, ::, :?, sizeof**.

Для перегрузки операции задаётся **оператор-функция** (операторная функция), которая является либо членом класса, либо дружественной классу, для которого она задана:

тип *имя_класса***::operator #(**список параметров**){**
// **…**
}

Здесь вместо знака # ставится знак перегружаемой операции.

Существуют некоторые ограничения на перегрузку операций: нельзя менять приоритет операций; нельзя менять число операндов операции; нельзя создавать новые операции; оператор-функции не могут иметь аргументов по умолчанию.

Бинарная операция перегружается как функция-член класса с одним параметром или как дружественная функция с двумя параметрами. Один из этих параметров должен быть либо объектом класса, либо ссылкой на объект этого класса.

Унарные операции имеют один операнд, поэтому унарная операция класса перегружается как функция-член без параметров или как дружественная функция с одним параметром. Этот параметр должен быть или объектом, или ссылкой на объект этого класса.

Ссылка на некоторый объект может рассматриваться как указатель, который при работе с ним всегда разыменовывается. Для определения ссылки применяется унарный оператор **&**. Ссылка не создаёт копию объекта, а лишь является другим именем объекта. Чаще всего ссылки используются для передачи аргументов в функции.

Операция присваивания перегружается для случая, когда нежелательно побитовое копирование, но только как функция-член класса. Функция **operator=()** возвращает ****this,*** так как функция возвращает объект, которому присваивается значение, т. е. левый операнд в операции изменяется. Это позволяет выполнить любую цепочку присваиваний (**ob1 = ob2 = ob3**).

Перегрузка операций ввода-вывода “>>” и “>>” в **C++** предусмотрена для выполнения ввода-вывода объектов класса.

Для вывода объектов создаётся оператор-функция (функция вставки символов в объект), общий формат которой имеет вид:

```
ostream& operator<<(ostream &stream, имя_класса obj){
   // …
  return stream;
}
```

Первый параметр в функции является ссылкой на объект типа **ostream**. Это означает, что поток ***stream*** должен быть потоком вывода. Второй параметр получает выводимый объект (он, если это нужно, тоже может быть параметром-ссылкой). Функция возвращает ссылку на объект типа **ostream** что позволяет использовать оператор << в ряде последовательных выражений вывода (**cout<<ob1<<ob2<<ob3**).

Для ввода объектов создаётся оператор-функция (функция извлечения символов из потока), формат которой имеет следующий вид:

istream& **opepator>>(**istream &***stream,*** имя_класса &obj)**{**
 //…
 return ***stream*****;**
}

Первый параметр – ссылка на объект типа **istream**, т. е. поток ***stream*** должен быть потоком ввода. Второй параметр – ссылка на объект, получающий вводимую информацию. Функция возвращает ссылку на объект типа **istream,** что позволяет использовать оператор >> в ряде последовательных выражений ввода (**cin>>ob1>>ob2>>ob3**).

Следует помнить, что пользовательские функции ввода-вывода не должны быть членами класса, для ввода-вывода данных которого они предназначены. Они могут быть дружественными функциями этого класса или просто независимыми функциями.

Постановка задания

Определить класс, который должен включать в себя конструктор, деструктор, операторные функции, обеспечивающие перегрузку стандартных операций для работы с объектами данного класса. Для указанного класса также перегрузить операции << и >> для ввода и вывода объектов.

Написать программу, демонстрирующую концепцию перегрузки операций.

Примеры выполнения задания

```
// Пример 5.1. Перегрузка ввода-вывода.
// Нахождение наименьшего делителя числа.
#include <iostream.h>
class Factor{                  // определение класса Factor
  int num;                     // число
  int min_del;                 // наименьший делитель
public:
  Factor();                    // конструктор по умолчанию
  void delit();
  friend ostream& operator<<(ostream &stream, Factor ob);
  friend istream& operator>>(istream &stream, Factor &ob);
};
```

```
void Factor::delit(){          // определение функции delit()
  int n;
  int i = num;
  for(n = 2; n  <  (i / 2); n++)
    if(i % n == 0) break;
    if(n < (i / 2)) min_del = n;
    else min_del = 1;
}
// оператор-функция для ввода объектов
istream& operator>>(istream &stream, Factor &ob){
  cout<<"Vvedi chislo:";
  stream>>ob.num;
  return stream;
}
// оператор-функция для вывода объектов
ostream& operator<<(ostream &stream, Factor ob){
  stream <<endl<< ob.min_del<< " min delitel chisla  ";
  stream << ob.num << '\n';
  return stream;
}
int main(){              // функция main()
  Factor ob;             // создание объекта ob типа Factor
  cin>>ob;               // ввод объекта ob
  ob.delit();            // вызов функции delit()
  cout<<ob;              // вывод объекта ob
 system("pause");
  return 0
}
```

Результат:

```
Vvedi chslo:12
2 min delitel chisla 12
```

// **Пример 5.2.** Перегружается **операция +** для сложения объектов,
// члены класса – одномерный массив и его размерность.
// Объект в функцию **operator+()** передаётся по ссылке, возвращается
// по значению. Возвратить ссылку на объект функция не может,
// т. к. объект **temp** – локальный

```
#include <iostream.h>
const n_max = 10;
```

```
class Mas{                          // определение класса Mas
  int a[n_max];                     // объявление массива
  int n;                            // размерность массива
public:
  Mas();                            // конструктор по умолчанию
  Mas(int k);                       // конструктор с параметром
  Mas operator+(Mas &ob);           // функция operator+()
  friend  ostream& operator<<(ostream &stream , Mas &ob);
  friend  istream& operator>>(istream &stream , Mas &ob);
};
Mas::Mas(int k){                    // определение конструктора
  n = k;                            // инициализация n
}

// оператор-функция ввода объекта
istream& operator>>(istream &stream , Mas &ob){
  int i;
  for(i = 0; i < ob.n; i++){
    cout<<"Vvedi a["<<i<<"] ";
    stream>>ob.a[i];
  }
  return stream;
}

// оператор-функция вывода объекта
ostream& operator<<(ostream &stream , Mas &ob){
  int i;
  for(i = 0; i < ob.n; i++)
    stream<<ob.a[i]<<" ";
  stream<<endl;
  return stream;
}

Mas Mas::operator+(Mas ob){         // функция operator+()
  Mas temp(n);
  for (int i = 0; i < n; i++)
    temp.a[i] = a[i] + ob.a[i];
  return temp;                      // возврат объекта temp
}

int main(){                         // функция main()
  int n1;
```

```
  cout<<"Vvedi razmernost massiva: "; cin>>n1;
  Mas ob1(n1), ob2(n1), ob3(n1);              // создание объектов
  cout<<"Vvedi ob1:"<<endl;  cin>>ob1;        // ввод объекта ob1
  cout<<"Vvedi ob2:"<<endl;  cin>>ob2;        // ввод объекта ob2
  cout<<"ob1 = ";   cout<<ob1;                // вывод объекта ob1
  cout<<"ob2 = ";   cout<<ob2;                // вывод объекта ob2
  ob3 = ob1 + ob2;                            //  ob1 + ob2
  cout<<"ob3 = ";   cout<<ob3;                // вывод объекта ob3
 system("pause");   return 0;
}
```

Результат:

```
Vvedi razmernost massiva: 5
ob1= 1 2 3 4 5    ob2= 2 3 4 5 6    ob3= 3 5 7 9 11
```

```
// Пример 5.3. Перегружается операция + для сложения объектов,
// членами которых являются динамические одномерные массивы.
// Объект в функцию operator+() передаётся по ссылке, возвращается по
// значению, поэтому объект в функцию operator=() передаётся по
// значению. Используется конструктор копии.

#include <iostream.h>
class Mas{
  int *a;                    // указатель на тип int
  int n;                     // размерность массива
public:
  Mas();                     // конструктор по умолчанию
  Mas(int k);                //конструктор с параметром
  Mas(const Mas &ob);        // конструктор копии
  ~Mas();                    // деструктор
  Mas operator+(Mas& ob);              // функция operator+()
  Mas& operator=(Mas ob);              // функция operator=()
  friend  ostream& operator<<(ostream& stream , Mas &ob);
  friend  istream& operator>>(istream& stream , Mas &ob);
};

Mas::Mas(int k){                   //  конструктор
  n = k;                           // инициализация n
  a = new int[n];                  // динамическая память для массива
}
```

```
Mas::Mas(const Mas &ob){        //  конструктора копии
  n = ob.n;
  a = new int[n];
  for (int i = 0; i < n; i++)
    a[i] = ob.a[i];
}
istream& operator>>(istream& stream , Mas& ob){
  for(int i = 0; i < ob.n; i++){
    cout<<"Vvedi a["<<i<<"] ";   stream>>ob.a[i];
  }
  return stream;
}
ostream& operator<<(ostream& stream , Mas& ob){
  for(int i = 0; i < ob.n; i++)
    stream<<ob.a[i]<<" ";
  stream<<endl;
  return stream;
}
Mas& Mas::operator=(Mas ob){          // функция operator=()
  n = ob.n;
  for(int i = 0; i < n; i++)
    a[i] = ob.a[i];
  return *this;
}
Mas Mas::operator+(Mas& ob){          // функция operator+()
  Mas temp(n);
  for (int i = 0; i < n; i++){
    temp.a[i] = a[i] + ob.a[i];
 return temp;                         // возврат объекта temp
}
Mas::~Mas(){                          // деструктор
  delete []a;
}

int main(){                           //функция main()
  int n1;
  cout<<"Vvedi razmernost massiva: "; cin>>n1;
  Mas ob1(n1),ob2(n1),ob3(n1); // создание объектов ob1, ob2, ob3
  cout<<"Vvedi ob1"<<endl;  cin>>ob1;
  cout<<"Vvedi ob2"<<endl;  cin>>ob2;
```

```
  cout<<"ob1= ";  cout<<ob1;            // вывод объектов
  cout<<"ob2= ";  cout<<ob2;
  ob3 = ob1 + ob2;                      // сложение объектов
  cout<<"ob3= ";  cout<<ob3;
  system("pause");
  return 0;
}
```

// **Пример 5.4.** Перегружаются операции **"+"** и **"!"** для объектов-матриц:
// **"+"** – для сложения объектов-матриц;
// **"!"** – для определения кол-ва строк матрицы, не содерж. нулевых элементов.
// Перегружаются операwbb **">>"** и **"<<"** для ввода/вывода объектов.
// Объект передаётся в оператор-функцию ***operator*+()** по ссылке,
// возвращается объект из функции ***operator*+()** по значению.

```
#include <iostream.h>
#include <stdlib.h>
const n_max = 10,
      m_max = 10;
class Matrix{                                   // определение Matrix
  int a[n_max][m_max];                          // объявление матрицы
  int n;   int m;                               // размерности матрицы
public:
  Matrix(int k, int l);
  Matrix operator+(Matrix &ob);                 // функция operator+()
  int operator!();                              // функция operator!()
  friend  ostream& operator<<(ostream& stream , Matrix &ob);
  friend  istream& operator>>(istream& stream , Matrix &ob);
};
Matrix::Matrix(int k, int l){                   // конструктор
  n = k;  m = l;
}
Matrix Matrix::operator+(Matrix &ob){           // функция operator+()
  Matrix temp(n, m);
  for(int i = 0; i < n; i++)
    for(int j = 0; j < m; j++)
      temp.a[i][j] = a[i][j] + ob.a[i][j];
  return temp;
}
int Matrix::operator!(){                        //функция operator!()
  int k1 = 0;
```

```
  for(int i = 0; i < n; i++)
    for(int j = 0; j < m; j++)
      if(a[i][j] == 0){k1++;  break;}
  }
  int kol = n-k1;
  return kol;
}
istream& operator>>(istream& stream , Matrix &ob){       // ввод
  for(int i = 0; i<ob.n; i++)
    for(int j = 0; j<ob.m; j++){
      cout<<"a["<<i<<" "<<j<<"] ";   stream>>ob.a[i][j];
    }
  return stream;
}
ostream& operator<<(ostream& stream , Matrix &ob){     // вывод
  for(int i = 0; i < ob.n; i++){
    for(int j = 0; j < ob.m; j++){
      stream.width(4);
      stream<<ob.a[i][j];
    }
    stream<<endl;
  }
  stream<<endl;
  return stream;
}
int main(){                                          // функция main()
  int n, m;
  cout<<"***OVERLOAD***"<<endl;
  cout<<"Vvedi razmernost matrix: ";cin>>n>>m;
  Matrix ob1(n, m), ob2(n, m);                       // объекты ob1, ob2
  cout<<"Vvedi matrix 1"<<endl;  cin>>ob1; // ввод ob1
  cout<<ob1;                                         // вывод ob1
  cout<<"Vvedi matrix 2"<<endl; cin>>ob2;  // ввод ob2
  cout<<ob2;                                         // вывод ob2
  Matrix ob3(n, m);                                  // создание ob3
  ob3 = ob1 + ob2;                                   // ob3 = ob1 + ob2
  cout<<"Summa of matrix..."<<endl;    cout<<ob3;
  int kol;
  kol = !ob1;                                        // operator!() для ob1
  cout<<"kol ne null_strok  ob1="<<kol<<endl;
```

```
  kol=!ob2;                                   // operator!() для ob2
  cout<<"kol ne null_strok ob2="<<kol<<endl;
  system("pause");
  return 0;
}
```

// **Пример 5.5**. Перегружается **операция "+"** для объектов-матриц,
// использующих динамическую память. В функцию ***operator*+()**
// объект передаётся по ссылке, возвращается объект по значению,
// поэтому в функцию ***operator*=**() объект передаётся по значению.
//Используется конструктор копии.

```
#include <iostream.h>
#include <stdlib.h>
class Matrix{                       // определение класса Matrix
  int **a;                          //указатель на массив указателей
  int n, m;
public:
  Matrix(int k, int l );
  ~Matrix();
  Matrix(const Matrix &ob);         //конструктор копии
  Matrix operator+(Matrix &ob);     //функция operator+()
  Matrix& operator=(Matrix ob);     //функция operator=()
  friend  ostream& operator<<(ostream &stream , Matrix &ob);
  void vvod();                      //функция ввода матрицы
};
Matrix::Matrix(int k, int l){       // конструктор
  n = k; m = l;
 a = new int *[n];
  for(int i = 0; i < n; i++)
    a[i] = new int[m];
}
Matrix::~Matrix(){                  // деструктор
 for(int i = 0; i < n; i++)
  delete []a[i];
 delete []a;
}
Matrix::Matrix(const Matrix &ob){   // конструктор копирования
  a = new int*[ob.n];
  for(int i = 0; i < ob.n; i++)
```

```
    a[i] = new int[ob.m];
  for(i = 0; i < ob.n; i++)
    for(int j = 0; j < ob.n; j++)
      a[i][j] = ob.a[i][j];
  n = ob.n;    m = ob.m;
}
Matrix Matrix::operator+(Matrix &ob){        // функция operator+()
  Matrix temp(n,m);
  for(int i = 0; i < n; i++)
    for(int j = 0; j < m; j++)
      temp.a[i][j] = a[i][j]+ob.a[i][j];
  return temp;
}
Matrix& Matrix::operator=(Matrix ob){        // функция operator=()
  for(int i = 0; i < n; i++)
    for(int j = 0; j < m; j++)
      a[i][j] = ob.a[i][j];
  return *this;
}
ostream &operator<<(ostream &stream , Matrix &ob){     // вывод
  for(int i = 0; i < ob.n; i++){
    for(int j = 0; j < ob.m; j++){
      stream.width(4);
      stream<<ob.a[i][j];
    }
  stream<<endl;
  }
  return stream;
}
void Matrix::vvod(){        // функция ввода vvod()
  cout<<"Vvedi matrix"<<endl;
  for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++){
      cout<<"a["<<i<<" "<<j<<"] ";cin>>a[i][j];
    }
}
int main(){
  int n, m;
  cout<<"Vvedi razmernost matrix: "; cin>>n>>m;
```

```
  cout<<"***OVERLOAD***"<<endl;
  Matrix ob1(n, m), ob2(n, m);
  ob1.vvod();                    // ввод объекта ob1
  cout<<ob1;                     // вывод объекта ob1
  ob2.vvod();                    // ввод объекта ob2
  cout<<ob2;                     // вывод объекта ob2
  Matrix ob3(n, m);
  ob3 = ob1 + ob2;               // ob3 = ob1 + ob2
  cout<<"Summa of matrix== "<<endl;    cout<<ob3;
 system("pause");  return 0;
}
```

Варианты контрольных заданий

1. Задан вектор в ***n***-мерном евклидовом пространстве. Реализовать:

   a) сложение векторов;
   b) вычитание векторов;
   c) скалярное произведение векторов;
   d) векторное произведение векторов;
   e) умножение вектора на скаляр.

2. В британском формате дата задаётся как число/месяц/год. Реализовать с учётом (и без учёта) високосных годов:

   a) сложение даты и заданного количества дней;
   b) вычитание из даты заданного количества дней;
   c) сложение двух дат;
   d) вычитание из одной даты другой;
   e) вычисление числа дней, прошедших между двумя датами.

3. Время задаётся в формате час, минута, секунда. Реализовать следующие операции (учесть переход через 24 часа):

   a) сложение времени и заданного количества секунд;
   b) вычитание из времени заданного количества секунд;
   c) сложение двух моментов времени;
   d) вычитание из одного момента времени другого;
   e) подсчёт числа секунд между двумя моментами времени, лежащими в пределах одних суток.

4. Комплексное число задаётся своим модулем и углом (например, число 10 * (**cos**(π / 6) + ***i*** *__sin(π/6__)) задаётся парой (10, **π** /6)). Реализовать:

   a) сложение чисел, вычитание чисел;

b) произведение чисел, деление чисел;
c) возведение в целочисленную степень;
d) извлечение квадратного корня.

5. Комплексное число задаётся своей вещественной и мнимой частями (например, 5 +3i задаётся парой (5, 3)). Реализовать:

a) сложение чисел, вычитание чисел;
b) произведение чисел, деление чисел ;
c) возведение в целочисленную степень;
d) извлечение квадратного корня.

6. Определить класс для работы с объектом “полином”. Реализовать следующие операции:

a) сложение двух полиномов, вычитание двух полиномов;
b) умножение полинома на число;
c) вычисление значения полинома в точке ***x*;**
d) дифференцирование полинома;
e) интегрирование полинома.

7. Определить класс для работы с объектом “ множество целых чисел”. Реализовать следующие операции:

a) объединение двух множеств;
b) пересечение двух множеств;
c) разность двух множеств;
d) добавление элемента во множество;
e) удаление элемента из множества.

8. Определить класс для работы с объектом “рациональная дробь” вида ***m*/*n***. Реализовать следующие операции:

a) сложение и вычитание двух дробей;
b) умножение двух дробей; деление двух дробей;
c) приведение дроби к несократимому виду;
d) вывод дроби в виде ***m*/*n*;**
e) сравнение двух дробей.

9. Определить класс для работы с квадратной матрицей. Реализовать следующие операции:

a) сложение матриц, вычитание матриц;
b) умножение матрицы на скаляр, деление матрицы на скаляр;
c) транспонирование матрицы;
d) умножение матрицы слева на её транспонированную матрицу.

10. Определить класс для работы с квадратной матрицей. Получить величину ***к*** одним из нижеуказанных способов и выполнить умножение матрицы на значение ***к***:

a) ***к*** – количество строк матрицы, не содержащих ни одного нулевого элемента;
b) ***к*** – количество столбцов матрицы, содержащих хотя бы один нулевой элемент;
c) ***к*** – количество отрицательных элементов в тех строках матрицы, которые содержат хотя бы один нулевой элемент;
d) ***к*** – номер строки матрицы, в которой находится наибольшая по длине возрастающая последовательность;
e) ***к*** – сумма всех чётных элементов матрицы, стоящих после нечётных элементов;
f) ***к*** – максимальный элемент среди элементов строк матрицы, упорядоченных по возрастанию или убыванию.

11. Определить класс для работы с квадратной матрицей. Сформировать вектор ***B*** одним из нижеуказанных способов и умножить его на заданную матрицу:

a) ***B*** – первая из строк матрицы, не содержащих ни одного положительного элемента;
b) ***B*** – строка матрицы, среднее арифметическое элементов которой минимально;
c) ***B*** – последний из столбцов матрицы, в которых имеются одинаковые элементы;
d) ***B*** – последний из столбцов матрицы, содержащих минимальное количество различных элементов;
e) ***B*** – первая из строк матрицы с максимальным количеством знакочередующихся элементов;
f) каждый элемент вектора ***B*** равен количеству отрицательных элементов соответствующей строки матрицы, кратных 3 или 5;
g) каждый элемент вектора ***B*** равен количеству максимальных элементов соответствующей строки матрицы;
h) каждый элемент вектора ***B*** равен количеству минимальных элементов соответствующего столбца матрицы.

12. Определить класс для работы с квадратной матрицей. Получить новую матрицу путём преобразования заданной матрицы одним из нижеуказанных способов и сложить её с исходной матрицей:

a) поменять местами максимальные элементы нижнего и верхнего треугольников матрицы относительно главной диагонали;

b) в каждой строке матрицы поменять местами первый максимальный и последний минимальный элементы;
c) элементы строк матрицы, содержащих хотя бы одно простое число, расположить в порядке возрастания;
d) элементы столбцов матрицы, не содержащих нулевых элементов, расположить в порядке убывания;
e) элементы каждой строки матрицы, кратные заданному числу ***p***, переставить в начало строки и расположить их в порядке возрастания;
f) положительные элементы каждого столбца матрицы переставить в конец столбца и расположить их в порядке убывания.

13. Определить класс для работы с квадратной матрицей. Получить новую матрицу путём преобразования заданной матрицы одним из нижеуказанных способов и умножить её на исходную матрицу:

a) переставить строки матрицы в порядке возрастания элементов первого столбца;
b) заменить максимальные элементы строк матрицы, не содержащих чётных элементов, на обратные величины;
c) элементы каждого столбца матрицы, стоящие за первым максимальным элементом столбца, расположить в порядке возрастания;
d) элементы столбцов матрицы, не содержащих положительных элементов, заменить суммой их цифр;
e) заменить нулями элементы матрицы, стоящие на пересечении строк и столбцов, содержащих хотя бы один нулевой элемент.

14. Определить класс для работы с «**Булевой матрицей**» размерности (***n***, ***m***). Реализовать следующие операции:

a) логического сложения матриц;
b) логического умножения матриц;
c) инверсии матриц;
d) получения количества строк матрицы, не содержащих ни одного нулевого элемента.

15. Известно, что в каждой строке и каждом столбце квадратной матрицы имеется единственный отрицательный элемент; переставить строки матрицы так, чтобы отрицательные элементы находились на главной диагонали.

ЗАДАНИЕ 6. НАСЛЕДОВАНИЕ

Цель задания – получение практических навыков создания и построения иерархии классов, изучение виртуальных функций и использование их для реализации динамического полиморфизма.

Постановка задания

Определить базовый класс и наследующие его производные классы.

Для проверки функционирования созданных классов написать программу, использующую эти классы. Предусмотреть передачу аргументов конструкторам базового класса, использование виртуальных функций для реализации динамического полиморфизма.

В заданиях перечисляются только обязательные члены-переменные и функции-члены (члены и методы) класса. Можно задавать дополнительные члены и методы, если они не отменяют обязательные и обеспечивают дополнительные удобства при работе с данными классами.

Пример выполнения задания

```
// Пример 6.1. Определить базовый класс Строка, предусмотрев ввод,
// вывод строки, получение длины строки, очистку строки.
// Определить производный класс Десятичная Строка. Предусмотреть
// ввод, вывод десятичных чисел,  проверку, является ли введённая
// строка десятичным числом, если нет - присвоить ей нулевое значение,
// сравнение двух чисел.
      #include<iostream.h>
      #include<conio.h>
      #include<string.h>
      #define N 100
      class Stroka {            // определение базового класса Stroka
        protected:
          char str[N];
      public:
          Stroka (){str[0]='\0'; }
          virtual void create();              // создание строки
          void show();                // вывод строки
          int length();               // определение длины строки
          void clear();               // очистка строки
        };
      class DecStroka : public Stroka { // опр. производного класса DecStroka
        public:
          void create();                      // создание строки
          bool isnumber();                    // проверка строки на число
          int operator==(DecStroka );         // сравнение двух строк
```

```
};
int menu();                              //  меню

void Stroka::create() {            // создание строки базового класса
  cout<<"Input your stroky: ";
  flushall();    gets(str);
}
void Stroka::show(){               // вывод строки
  int i=0;
  cout<<"Your stroka: (";
  cout<<str;    cout<<')'<<endl;
}
int Stroka::length(){                    // определение длины строки
  int i=0;
  while(!str[i]=='\0')    i++;
  return i;
}
void Stroka::clear(){              // очистка строки
   *str='\0';
   cout<<"Your stroka ochichena "<<endl;
}
  void DecStroka::create(){      // создание строки производного класса
   char s[N];      char c;
   cout<<"Do you want your number will be positive or negative ? (+/-) ";
   cin>>c;
   str[0]=c;    str[1]='\0';
   cout<<"Input your decstroky: ";
   flushall();
   gets(s);      strcat(str, s);
  }
 bool DecStroka::isnumber(){  // проверка строки на число
  bool pr(true);
  int k=0;     int l(0);
  if(str[0]=='+'||str[0]=='-')  k=1;
  l=length();                            // вызов length() базового класса
  for(int j=k;j<l;j++) {
    if(str[j]>='0'&& str[j]<='9') pr=true;
    else{      pr=false;      break;    }
}
  if(!pr) {    cout<<"Your stroka isn't a number "<<endl;
     clear(); }
  else  cout<<"Your stroka is a number "<<endl;
  show();                                // вызов show() базового класса
  return pr;
}
int DecStroka::operator==(DecStroka ob){  // сравнение двух строк
  return !strcmp(str,ob.str);
```

```
}
int main (){
  int m, l=0;
  Stroka *p;                          // указатель на базовый класс Stroka
  Stroka ob;                          // объект класса Stroka
  DecStroka d_ob1,d_ob2,d_ob3,d_ob4;
  p=&ob;          // указатель установлен на объект базового класса
  while (1){
   m=menu();
   switch (m){
    case 1: clrscr();
            p->create();            getch (); break;
    case 2: clrscr();
            ob.show();             getch(); break;
    case 3: clrscr();
            l = ob.length();    cout<<" dlina  stroki = ";     cout<<l;
            getch(); break;
    case 4: clrscr();
            p->clear();             getch(); break;
    case 5: clrscr();
            p=&d_ob1;  // указатель установлен на объект производного класса
            p->create();            getch(); break;
    case 6: clrscr();
            d_ob1.show();          getch(); break;
    case 7: clrscr();
            d_ob1.isnumber();    getch(); break;
    case 8: clrscr();
            cout<<"Input two number "<<endl<<endl;
            d_ob1.create();           d_ob2.create();
            if (d_ob1.isnumber()&& d_ob2.isnumber())
             if (d_ob1==d_ob2) cout<<"Your chicla ravnu "<<endl;
             else  cout<<"Your chicla ne ravnu "<<endl;
            else    cout<<"Your stroka isn't a number "<<endl;
            getch(); break;
    case 9: clrscr();          exit(1); break;
   }
  }
  return 0;
 }
int menu(){                           // меню
  int m;
  do{
   clrscr();
   cout<<" 1: Sozdat stroky "<<endl;
   cout<<" 2: Show stroky "<<endl;
   cout<<" 3: Opredelit dliny stroki"<<endl;
   cout<<" 4: Ochistit stroky "<<endl;
```

```
   cout<<" 5: Sozdat dec_stroky "<<endl;
   cout<<" 6: Show dec_stroky "<<endl;
   cout<<" 7: Opredelit if your sroka is a number "<<endl;
   cout<<" 8: Proverit dva chisla na ravenstvo "<<endl;
   cout<<" 9: Exit "<<endl;
   cin>>m;
  }while(m<1 || m>9);
 return m;
}
```

Варианты заданий

1. Создать абстрактный класс **Работник фирмы** и производные классы **Менеджер**, **Администратор**, **Программист**.

2. Создать базовый класс **Домашнее животное** и производные классы **Собака**, **Кошка**, **Попугай**. С помощью конструктора установить имя каждого животного и его характеристики.

3. Создать базовый класс **Садовое дерево** и производные классы **Яблоня**, **Вишня**, **Груша** и другие. С помощью конструктора автоматически установить номер каждого дерева.
Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

4. Определить классы и их иерархию, продемонстрировать использование введённых конструкций при работе **Учащийся: Школьник**, **Студент**.

5. Определить классы и их иерархию , продемонстрировать использование введённых конструкций при работе с **Музыкальный инструмент: Ударный**, **Струнный**, **Духовой**.

6. Определить классы и их иерархию, продемонстрировать использование введённых конструкций при работе с **Человек: Студент, Преподаватель.**

7. Определить базовый класс для работы с прямоугольными матрицами, предусмотрев следующие операции:

a) сложение матриц;
b) вычитание матриц;
c) умножение матрицы на скаляр;
d) деление матрицы на скаляр;
e) перестановка строк матрицы по заданному вектору транспозиции.

В производном классе переопределить указанные операции для квадратных матриц, добавив следующие операции:

a) транспонирование матрицы;

b) умножение матриц;
c) умножение матрицы слева на её транспонированную матрицу.

8. Определить базовый класс **Vector**, предусмотрев следующие операции:

a) проверку двух векторов на ортогональность;
b) определение количества единиц в векторе;
c) определение позиции самой левой единицы в векторе.

В производном классе **BoolVector** (компоненты вектора принимают значения из множества {0,1}) переопределить указанные операции для **Vector**, добавив следующие операции:

a) поразрядная конъюнкция (&);
b) поразрядная дизъюнкция (|);
c) поразрядная инверсия (~);
d) присваивание (=).

9. Определить базовый класс **Vecto**r, предусмотрев следующие операции:

a) проверка двух векторов на ортогональность;
b) определение количества нулевых компонент.

В производном классе **ТРОИЧНЫЙ ВЕКТОР** (компоненты вектора принимают значения из множества {0,1,2}) переопределить указанные операции для **Vector**, добавив следующие операции:

a) присваивание (=);
b) поразрядная конъюнкция двух не ортогональных векторов (&): 0&0=0, 1&1=1, 2&2=2, 0&2=0, 2&0=0, 1&2=1, 2&1=1;
c) индексирование ([]).

10. Определить базовый класс **МАТРИЦА** для работы с целочисленной матрицей, предусмотрев следующие операции:

a) получение числа единиц в матрице;
b) удаление повторяющихся строк.

В производном классе **БУЛЕВА МАТРИЦА** переопределить указанные операции, добавив следующие операции:

a) поэлементная конъюнкция двух матриц (&);
b) поэлементная дизъюнкция двух матриц (|);
c) произведение двух матриц;
d) присваивание (=).

11. Определить базовый класс **Символ**, предусмотрев следующие операции:

a) ввод, вывод символов;
b) перевод символов из нижнего регистра в верхний;
c) сравнение символов.

В производном классе **Строка** переопределить указанные операции, добавив следующие операции:

a) получение длины строки;
b) определить количество символов в самом длинном слове;
c) определить слова с заданным количеством символов и вывести их в алфавитном порядке;
d) определить и вывести в алфавитном порядке все слова, начинающиеся и заканчивающиеся гласной буквой.
e) следующие операции:

12. Определить базовый класс **Символ**, предусмотрев следующие операции:

a) ввод, вывод символов;
b) перевод символов из нижнего регистра в верхний;
c) сравнение символов.

В производном классе **Строка** переопределить указанные операции, добавив следующие операции:

a) получение длины строки;
b) определить количество символов в самом коротком слове;
c) определить количество символов '**a**' в последнем слове строки;
d) определить количество слов, в которых содержится хотя бы один заданный символ;
e) определить количество слов, оканчивающихся гласной буквой;
f) определить в каждом слове количество символов, отличных от букв;
g) определить порядковый номер заданного слова и количество символов в нём;

13. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки.
c) определение первого вхождения заданного символа.

В производном классе **Строка_Идентификатор** переопределить указанные операции, добавив следующие операции:

a) определение, является ли введённая строка идентификатором;
b) перевод всех символов строки_идентификатора в верхний регистр;
c) перевод всех символов строки_идентификатора в нижний регистр;

14. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки;

c) проверку строк на равенство.

В производном классе **Битовая_Строка** переопределить указанные операции, добавив следующие операции:

a) ределение, является ли строка битовой строкой;
b) изменение знака битовой строки на противоположный (перевести число в дополнительный код);
c) сложение битовых строк.

15. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки;
c) проверку строк на равенство.

В производном классе **Десятичная_Строка** переопределить указанные операции, добавив следующие операции:

a) определение, является ли введённая строка десятичным числом;
b) сложение десятичных чисел;
c) вычитание десятичных чисел.

16. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки;
c) проверку строк на равенство.

В производном классе **Комплексное_число** (Строка состоит из двух полей, разделённых символом **i**. Каждое из полей может содержать только символы десятичных цифр и символы – или +. Символы – и + могут находиться только в первой позиции числа, причём символ + может отсутствовать, в этом случае число считается положительным. Содержимое данной строки рассматривается как комплексное число, например, 35i14 ,-4i100, +3-i21) переопределить указанные операции, добавив следующие операции:

a) определение, представляет ли введённая строка комплексное число;
b) сложение комплексных чисел;
c) вычитание комплексных чисел;
d) умножение комплексных чисел;
e) проверка чисел на равенство.

17. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки;

c) определение количества слов в строке.

В производном классе **Текст** переопределить указанные операции, добавив следующие операции:

a) в восклицательных предложениях текста найти и вывести без повторений все слова заданной длины;
b) из каждого предложения текста удалить слова, встречающиеся более одного раза;
c) определить в тексте самую длинную цепочку слов, начинающихся на гласную букву.
d) все слова каждого предложения текста вывести в алфавитном порядке;
e) в каждом предложении текста найти и вывести в алфавитном порядке все слова “перевёртыши” (одинаково читаются слева направо и справа налево).

18. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки;
c) определение количества слов в строке.

В производном классе **Текст** переопределить указанные операции, добавив следующие операции:

a) определить количество слов и предложений в тексте;
b) определить предложения с максимальным количеством слов;
c) найти в каждом предложении текста без повторений слова минимальной длины;
d) найти во всех вопросительных предложениях текста без повторений слова максимальной длины;
e) определить для заданного слова, сколько раз оно встречается в каждом предложении текста;

19. Определить базовый класс **Строка**, предусмотрев следующие операции:

a) ввод, вывод строки;
b) получение длины строки;
c) выделение слов в строке при нажатии произвольной клавиши.

В производном классе **Текст** переопределить указанные операции, добавив следующие операции:

a) поочерёдно выделить каждое предложение текста;
b) поочерёдно выделить каждое предложение текста, а в выделенном предложении – поочерёдно все слова;

c) поочерёдно выделить каждое предложение текста, а в выделенном предложении – все слова, в которых буквы расположены в алфавитном порядке;
d) поочерёдно выделить каждое предложение текста, а в выделенном предложении – поочерёдно заданное слово;
e) поочерёдно выделить каждое вопросительное предложение текста, а в выделенном предложении – поочерёдно все символы, отличные от букв и пробелов;

20. Определить базовый класс **Строка**, предусмотрев следующие операции:
a) ввод, вывод строки;
b) получение длины строки;
c) выделение слов в строке при нажатии произвольной клавиши.

В производном классе **Текст** переопределить указанные операции, добавив следующие операции:
a) выделить последнее предложение текста, а в выделенном предложении – поочерёдно все буквы ‘а’;
b) выделить каждое слово текста, содержащее максимальное количество символов;
c) поочерёдно выделить в тексте каждое слово, заданное слово;
d) поочерёдно выделить в тексте заданные слова, которые нужно поменять местами (заданные слова вводить с клавиатуры);
e) поочерёдно выделить все слова текста, содержащие цифры;
f) поочерёдно выделить в тексте все слова максимальной длины.

21. Определить базовый класс **Множество** и производный класс **Кольцо** (операции сложения и умножения, обе коммутативные и ассоциативные, связанные законом дистрибутивности. Сложение обладает обратной операций – вычитанием).
Ввести кольца целых чисел многочленов, система классов целых чисел, сравнимых по модулю. Кольцо является полем, если в нём определена операция деления, кроме деления на ноль. Рациональные числа, дробно рациональные функции.

22.Определить классы и их иерархию, продемонстрировать использование введённых конструкций при работе с **Меню**: **Горизонтальное, Вертикальное, Иерархическое.**

23.Определить классы и их иерархию, продемонстрировать использование введённых конструкций при работе с **Окно**: **Стековое, Слоёное, Всплывающее.**

24. Создать базовый класс **Polygon** . Класс должен содержать методы для рисования многоугольника, вычисления его периметра, нахождения

площади и др. Построить производный класс **Triangle**, который содержит также методы для нахождения точки пересечения медиан, длин медиан, длин биссектрис, координат точек пересечения биссектрис со сторонами треугольника, длин высот треугольника.

25. Создать абстрактный класс **Shape** для рисования плоских фигур. Построить производные классы **Square** (квадрат, который характеризуется координатами левого верхнего угла и длиной стороны), **Circle** (окружность с заданными координатами центра и радиусом), **Ellipse** (эллипс с заданными координатами верхнего угла описанного вокруг него прямоугольника со сторонами, параллельными осям координат, и малым и большим диаметрами), позволяющие рисовать указанные фигуры, а также передвигать их на плоскости.

26. Создать класс **Point** и производные от него классы **ColorPoint и Line**. На основе классов **ColorPoint и Line** создать класс **ColorLine**. Все классы должны иметь методы для установки и получения значений всех координат, а также изменения цвета и получения текущего цвета.

27. Расширить возможности стандартного класса **Time**, так чтобы была возможность выводить время дня: утро, вечер и т.д.

28. Расширить возможности стандартного класса **Date**, так чтобы была возможность выводить время года: зима, лето и т.д.

29. Расширить возможности класса **Annotation**, так чтобы была возможность выводить время и дату изменения аннотации.

30. Расширить возможности класса **Dictionary**, так чтобы была возможность выводить дату последнего изменения в словаре.

31. Расширить возможности класса **File**, так чтобы была возможность выводить время и дату создания файла.

32. Расширить возможности класса **Stack**, так чтобы была возможность выводить время последнего сеанса работы со стеком.

Оглавление