Работа 4.

Тема: Шаблоны. Конструктор копирования. Перегрузка операций

Задача: Создание шаблонного класса Матрица

Создать шаблонный класс Матрица. Размер матрицы m*n. Тип элементов задается через параметр шаблона. Память для матрицы выделяется динамически.

Элементы класса:

- переменная M типа «указатель на указатель». Эта переменная определяет матрицу. Память для матрицы будет выделяться динамически;
- переменные m, n – это размерность матрицы M;
- конструктор по умолчанию (без параметров);
- конструктор с двумя параметрами – создает матрицу размером m*n. В конструкторе выделяется память для столбцов и строк матрицы. Значение каждого элемента матрицы устанавливается в 0;
- конструктор копирования Matrix (Matrix &). Этот конструктор необходим для создания копии объекта-матрицы из другого объекта-матрицы;
- методы чтения/записи элементов матрицы GetM(), SetM();
- метод Print() – вывод матрицы;
- оператор копирования operator=(Matrix &). Этот оператор перегружает оператор присваивания = и предназначен для корректного копирования объектов, например, obj2=obj1;
- деструктор.

II. В качестве демонстрационного примера написать 2 варианта программы:

Вариант 1

```
int main()
{

  Matrix <int> M(3, 4);
  M.Print("M");

  // Заполнить матрицу значениями по
формуле
  int i, j;
  for (i = 0; i < 2; i++)
    for (j = 0; j < 3; j++)
    M.SetM(i, j, i + j);

  M.Print("M");

  Matrix < int > M1 = M; // вызов
конструктора копирования
M1.Print("M1");
```

Результат работы программы (для варианта 1):

```
Object: M
0     0     0     0
0     0     0     0
0     0     0     0
--------------------
Object: M
0     1     2     0
1     2     3     0
0     0     0     0
--------------
```

```
  Matrix < int > M2;
  M2 = M; // вызов оператора копирования
- проверка
  M2.Print("M2");

  Matrix < int > M3;
  M3 = M2 = M1 = M; // вызов оператора
копирования в виде "цепочки"
M3.Print("M3");
}
```

```
Object: M1
0     1     2     0
1     2     3     0
0     0     0     0
----------------------
Object: M2
0     1     2     0
1     2     3     0
0     0     0     0
- - - - - - - - - - - - - -
Object: M3
0     1     2     0
1     2     3     0
0     0     0     0
- - - - - - - - - - - - - -
```

<u>Вариант 2</u>

Такая же программа, как в варианте 1, но тип элементов матрицы - double:

```
int main()
{

  Matrix <double> M(3, 4);
  M.Print("M");

  // Заполнить матрицу значениями по
формуле
  int i, j;
  for (i = 0; i < 2; i++)
    for (j = 0; j < 3; j++)
      M.SetM(i, j, (i + j)*0.5);

  M.Print("M");

  Matrix <double> M1 = M; // вызов
конструктора копирования
M1.Print("M1");

  Matrix <double> M2;
  M2 = M; // вызов оператора копирования
- проверка
  M2.Print("M2");

  Matrix <double> M3;
```

Результат работы программы (для варианта 2):

```
Object: M
0     0     0     0
0     0     0     0
0     0     0     0
----------------------
Object: M
0     0.5    1     0
0.5    1     1.5    0
0     0     0     0
----------------------
Object: M1
0     0.5    1     0
0.5    1     1.5    0
0     0     0     0
----------------------
Object: M2
0     0.5    1     0
0.5    1     1.5    0
0     0     0     0
----------------------
```

M3 = M2 = M1 = M; // вызов оператора копирования в виде "цепочки"
M3.Print("M3");
}
	Object: M3
	0 0.5 1 0
	0.5 1 1.5 0
	0 0 0 0
	- - - - - - - - - - - - -