Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Севастопольский Государственный Университет

Институт информационных технологий и управления в технических системах

Кафедра Информационных систем

**ЛАБОРАТОРНАЯ РАБОТА №2**

По дисциплине: Объектно-ориентированное программирование

По теме: Простое наследование. Виртуальные функции.

Вариант

Выполнила:   
студентка ИС/б-25о группы  
Изосимина Ю.Ю.

Проверила:  
Тимофеева Т.И.

Севастополь 2015

1. ЦЕЛЬ РАБОТЫ

Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизма наследования и механизма виртуальных функций. Освоение особенностей отладки объектно-ориентированных программ.

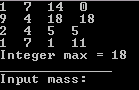
**2. ВАРИАНТЫ ЗАДАНИЙ**

Для заданной по варианту иерархии описать классы, конструкторы и деструктор, функции ввода и вывода информации на экран. Базовый класс определить как абстрактный, а заданную функцию — как чисто виртуальную в базовом классе и переопределить ее в остальных классах иерархии. Проиллюстрировать корректную работу виртуальных функций и механизма наследования.

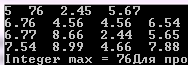
**Вариант *6***.

Создать абстрактный базовый класс Матрица с виртуальной функцией поиска максимального значения в массиве. Создать производные классы: матрица целых чисел (int\*\*), матрица символов (char \*\*) и матрица вещественных чисел (double\*\*).

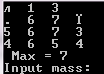
|  |
| --- |
| #include "stdafx.h"  #include <iostream>  using namespace std;  class Matrica {  public:  Matrica(){};  virtual void Poisk() = 0;  ~Matrica(){};  };  class Integer : public Matrica {  int max;  const int n = 4;  public:  int \*\* mas;  Integer() :Matrica(){  mas = new int\*[n];  for (int i = 0; i < n; i++){  mas[i] = new int[n];}}  void add(){  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  mas[i][j] = rand() % 20;  cout << mas[i][j] << " ";}  cout << endl;}}  virtual void Poisk(){  max = mas[0][0];  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  if (mas[i][j]>max)  max = mas[i][j];}}  cout << "Integer max = " << max; }};  class Char: public Matrica {  char max;  int n = 4;  char \*\* mas;  public:  Char() :Matrica(){  mas = new char\*[n];  for (int i = 0; i < n; i++)  mas[i] = new char[n];}  void add(){  cout << "Input mass: ";  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  cin >> mas[i][j];}  }}  virtual void Poisk(){  max = mas[0][0];  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  if (mas[i][j]>max)  max = mas[i][j];}}  cout << " Max = " << max << endl;}  void show(){  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  cout << mas[i][j] << " ";  }  cout << endl;}}  };  class Dbl: public Matrica {  double max;  int n = 4;  double \*\* mas;  public:  Dbl() :Matrica(){  mas = new double\*[n];  for (int i = 0; i < n; i++)  mas[i] = new double[n];  }  virtual void Poisk(){  max = mas[0][0];  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  if (mas[i][j]>max)  max = mas[i][j];  }  }  cout << "Integer max = " << max;  }  void add(){  cout << "Input mass: ";  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  cin >> mas[i][j];  }}}  void show(){  for (int i = 0; i < n; i++){  for (int j = 0; j < n; j++){  cout << mas[i][j] << " ";  }  cout << endl;}}};  int main()  {  setlocale(LC\_CTYPE, "RUSSIAN");  Integer matrix;  Char arr;  Dbl arr2;  matrix.add();  matrix.Poisk();  cout << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  arr.add();  system("pause");  system("cls");  arr.show();  arr.Poisk();  arr2.add();  system("pause");  system("cls");  arr2.show();  arr2.Poisk();  system("pause");  return 0;  } |

****

Матрица целых чисел (int\*\*)

****

Матрица вещественных чисел (double\*\*).

****

Матрица символов (char \*\*)

**Вывод:** В данной работе мы изучили механизм простого наследования и использование виртуальных функций. Также был рассмотрен механизм

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое простое наследование, для чего оно применяется? Язык C++ позволяет классу наследовать данные-члены и функции-члены одного или нескольких других классов. При этом новый класс называют производным классом (или классом-потомком). Класс, элементы которого наследуются, называется базовым классом (родительским классом или классом-предком) для своего производного класса. Наследование дает возможность некоторые общие черты поведения классов абстрагировать в одном базовом классе.

2. Что такое спецификатор доступа, для чего он нужен при наследовании? Спецификатор доступа — это одно из ключевых слов *private*, *protected* или *public*. Он не является обязательным и по умолчанию принимает значение *private* для классов и *public* для структур.

3. Привести пример переопределения функции и объяснить суть механизма. Чтобы переопределить функцию-член базового класса в производном классе, достаточно включить ее прототип в объявление этого класса и затем дать ее определение. #include <iostream.h>

class MyInt

{ protected:

int x;

public:

MyInt(int \_x) {x = \_x;}

MyInt( ){x = 0;}

int GetX( ){return x;} // Определим метод вывода поля для базового класса

void SetX(int \_x) {x = \_x;}

};

class IncMyInt: public MyInt

{ public:

IncMyInt(int \_x) : MyInt(\_x) {}

int GetX( ) {return ++x;} // Переопределим для наследника метод

// базового класса

void ShowX( ) {cout << GetX()<<' ';}

};

main( )

{

IncMyInt\* ptr;

ptr = new IncMyInt(10); // Создание объекта в куче

ptr->ShowX( );

delete ptr; //Удаление объекта в куче

return 0;

} Иногда возникает необходимость вызвать функцию-член базового класса, а не ее переопределенный вариант, это можно сделать с помощью операции расширения области видимости, применяемой в форме:

< имя\_класса> :: <имя\_функции>

4. Перечислить особенности описания конструкторов и деструкторов при наследовании. Конструкторы не наследуются, поэтому производный класс либо должен объявить свой конструктор, либо предоставить возможность компилятору генерировать конструктор по умолчанию. В связи с этим для построения конструктора производного класса применяется следующая конструкция:

<имя класса>([<список аргументов>]) : <имя класса предка>( [<список аргументов>])

В отношении деструкторов производных классов также действуют определенные правила. Деструктор производного класса должен выполняться раньше деструктора базового класса (иначе деструктор базового класса мог бы разрушить данные-члены, которые используются и в производном классе).

5.Что такое полиморфизм? Полиморфизм состоит в том, что с помощью одного и того же обращения к методу выполняются различные действия в зависимости от типа, на который ссылается указатель в каждый момент времени. *Полиморфные функции* - работают с объектом любого типа в пределах одной иерархии.

Объект, определенный через указатель или ссылку и содержащий Виртуальные методы, называется *полиморфным*

6. Что такое виртуальная функция? Когда она применяется? Привести пример. Виртуальные функции — это функции, которые гарантируют, что будет вызнана правильная функция для объекта безотносительно к тому, какое выражение используется для осуществления вызова Функции, у которых известен интерфейс вызова (то есть прототип), но реализация не может быть задана в общем случае, а может быть определена только для конкретных случаев, называются виртуальными (термин, означающий, что функция может быть переопределена в производном классе).

7. Что такое чисто виртуальная функция? Абстрактный класс? Для чего они нужны? Класс, содержащий хотя бы один чисто виртуальный метод, называется абстрактный. Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах. Абстрактный класс может использоваться только в качестве базового для других классов — объекты абстрактного класса создавать нельзя, поскольку прямой или косвенный вызов чисто виртуального метода приводит к ошибке при выполнении.

8. Что такое механизм позднего связывания?

В C++ реализован механизм позднего связывания, когда разрешение ссылок на метод происходит на этапе выполнения программы в зависимости от конкретного типа объекта, вызвавшего метод. Этот механизм реализован с помощью виртуальных методов.

Таблицу виртуальных методов (*vtbl*), в которой для каждого виртуального метода записан его адрес в памяти.

Виртуальным называется метод, ссылка на который разрешается на этапе выполнения программы