МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-2 «Прикладные информационные технологии»

А.А. МЕРСОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКОЙ РАБОТЫ № 8
Реализация полиморфизма

по дисциплине: **«Языки программирования»**
(наименование дисциплины)

Москва – 2021

УДК
ББК

Печатается по решению редакционно-издательского совета «МИРЭА – Российский технологический университет»

Мерсов А.А.

Методические указания по выполнению практической работы № 8 по языкам программирования / А.А. Мерсов– М.: МИРЭА – Российский технологический университет, 2021.

Методические указания предназначены для выполнения практической работы по дисциплине «Языки программирования» и содержит перечень вариантов практической работы, а также краткое изложение теоретического материала в форме пояснений к заданию на работу. Для студентов, обучающихся по направлениям 10.03.01, 10.05.02, 10.05.03, 10.05.04.

Материалы рассмотрены на заседании учебно-методической комиссии
КБ-2 Протокол №1 от «28» августа 2021 г.
и одобрены на заседании кафедры КБ-2.

зав. кафедрой КБ-2

к.т.н. _____ / О.В.Трубиенко /

УДК ББК

Содержание

Общие указания к выполнению практической работы

Практические работы выполняются с использованием персональных компьютеров. Указания по технике безопасности совпадают с требованиями, предъявляемыми к пользователю ЭВМ. Другие опасные факторы отсутствуют.

Цель практической работы

Цель работы: изучить одну из базовых концепций ООП – наследование классов в C++, заключающуюся в построении цепочек классов, связанных иерархически. Познакомиться с механизмом виртуальных функций..

Практическая работа предполагает выполнение задания разработке и тестированию программного обеспечения.

Наследование - механизм создания производного класса из базового. Т.е., к существующему классу можно что-либо добавить, или изменять его каким-либо образом для создания нового (производного) класса. Это мощный механизм для повторного использования кода. Наследование позволяет создавать иерархию связанных типов, совместно использующих код и интерфейс. Модификатор прав доступа используется для изменения доступа к наследуемым объектам в соответствии с правилами, указанными в таблице 1.

Таблица 1 – Доступ в классах при наследовании

Доступ в базовом классе	Модификатор прав доступа	Доступ в производном классе
private	private	не доступны
private	public	не доступны
protected	private	private
protected	public	protected
public	private	private
public	public	public

Ограничение на наследование

При определении производного класса не наследуются из базового:

1. конструкторы;
2. деструкторы;
3. операторы new, определенные пользователем;
4. операторы присвоения, определенные пользователем;
5. отношения дружественности.

Использование косвенной адресации с установкой указателей на базовый класс. Механизм косвенной адресации рассмотрим на примере:

```
class B
{
public:
int x;
B() {          // Конструктор по умолчанию
x = 4; }
};
class D : public B {      // Производный класс
public:
int y;
D()
{          // Конструктор по умолчанию
y = 5; }
};
```

```
void main(void)  {
D  d;// Конструктор класса D создает объект d
B *p; // Указатель установлен на базовый касс
p = &d;// Указатель p инициализируется адресом d
// косвенное обращение к объектам базового и производного классов
// «считываем их текущее состояние в переменные
int i = p -> x;  // Базовый класс виден напрямую
int j = ( ( D* ) p )p -> y;// Прямое преобразование указателя на D
// через переменные печатаем их текущее состояние
cout << " x_i= " << i << endl;
cout << " y_j= " << j << endl;
getch();
}
```

Виртуальная функция и механизм позднего связывания

Виртуальная функция объявляется в базовом или впроизводном классе и, затем, переопределяется в наследуемых классах. Совокупность классов (подклассов), в которых определяется и переопределяется виртуальная функция, называется полиморфическим кластером, ассоциированным с некоторой виртуальной функцией. В пределах полиморфического кластера сообщение связывается с конкретной виртуальной функцией-методом во время выполнения программы.

Обычную функцию-метод можно переопределить в наследуемых классах. Однако без атрибута virtual такая функция-метод будет связана с сообщением на этапе компиляции. Атрибут virtual гарантирует позднее связывание в пределах полиморфического кластера.

Часто возникает необходимость передачи сообщений объектам, принадлежащим разным классам в иерархии. В этом случае требуется реализация механизма позднего связывания. Чтобы добиться позднего связывания для объекта, его нужно объявить как указатель или ссылку на объект соответствующего класса. Для открытых производных классов указатели и ссылки на объекты этих классов совместимыми с указателями и ссылками на объекты базового класса (т.е. к объекту производного класса можно обращаться, как будто это объект базового класса). Выбранная функция-метод зависит от класса, на объект которого указывается, но не от типа указателя.

C++ поддерживает virtual функции-методы, которые объявлены в основном классе и переопределены в порожденном классе. Иерархия классов, определенная общим наследованием, создает связанный набор типов пользователя, на которые можно ссылаться с помощью указателя базового класса. При обращении к виртуальной функции через этот указатель в C++ выбирается соответствующее функциональное определение во время выполнения. Объект, на который указывается, должен содержать в себе информацию о типе, поскольку различия между ними может быть сделано динамически. Это особенность типична для ООП кода. Каждый объект “знает” как на него должны воздействовать. Эта форма полиморфизма называется чистым полиморфизмом.

В C++ функции-методы класса с различным числом и типом параметров есть действительно различные функции, даже если они имеют одно и тоже имя. Виртуальные функции позволяют переопределять в управляемом классе функции, введенные в базовом классе, даже если число и тип аргументов то же самое. Для виртуальных функций нельзя переопределять тип функции. Если две функции с одинаковым именем будут иметь

различные аргументы, C++ будет считать их различными и проигнорирует механизм виртуальных функций. Виртуальная функция обязательно метод класса.

Методический пример

Написать программу, реализующую использования виртуальной функции для получения данных у базового и производного класса

```
#include <iostream>
using namespace std;
class A
{
public:
	int x; int y;			// объявление  данных x и y  класса A
	int *ix,*iy;
	A() { x = 1; y = 2; }//инициализация данных в момент создания объектов
	void virtual setx(int z) { x = z; }// установка нового состояния x с помощью
виртуальн функции
	void virtual sety(int z) { y = z; }// установка нового состояния y с помощью
виртуальн функции
	int virtual getx() { return x; }   // получение состояния переменной x с помощью
виртуальн функции
	int virtual gety() { cout << "  A  " << endl; return y; }   // получение состояния
переменной y с помощью виртуальн функции
};
class B:public A {
public:
	int x; int y;			// объявление  данных x и y Класса B
	int *ix, *iy;
	B() { x = 3; y = 4; }//инициализация данных в момент создания объектов
	void virtual setx(int z) { x = z; }// установка нового состояния x с помощью
виртуальн функции
	void virtual sety(int z) { y = z; }// установка нового состояния y с помощью
виртуальн функции
	int virtual getx() { return x; }   // получение состояния переменной x с помощью
виртуальн функции
	int virtual gety() { cout << "  B  " << endl; return y; }   // получение состояния
переменной y с помощью виртуальн функции
};
void f(A* e) {  cout << e->getx() << "   " <<e->gety()<<endl; } //функция, принимающая
адрес на объект класса A
int main()
{
	A x1;
	B x2;
	f(&x1);
	f(&x2);

}
```

Задание к работе

Общая постановка. Программа должна содержать:

- базовый класс X, включающий два элемента x1, x2 типа int,
- конструктор с параметрами для создания объектов в динамической области памяти,
- деструктор,
- виртуальные методы просмотра текущего состояния и переустановки объектов базового класса в новое состояние.
- производный класс У, включающий один элемент y типа int ,

- конструктор с параметрами и списком инициализаторов, передающий данные конструктору базового класса,
- переопределенные методы просмотра текущего состояния объектов и их переустановки в новое состояние.

Варианты задания

Создать в производном классе метод Run, определяющий:

1. Сумму компонент классов
2. Произведение компонент классов
3. Сумму квадратов компонент классов
4. Значение x1+x2 – y
5. Значение (x1+x2)/y
6. Значение (x1+x2)*y
7. Значение x1*y+x2
8. Значение x1+x2*y
9. Произведение квадратов компонент класса

0. Значение x1*x2+y