**Раздел II. Технология программирования (C++)**

В этом разделе рассматриваются элементы технологии объектно-ориентированного программирования. В качестве базового языка используется язык С++, который рассматривается как некоторая надстройка над языком Си. То есть, язык С++ интерпретируется как расширение языка Си с поддержкой ООП (Object Oriented Programming, OOP).

**Основные понятия объектно-ориентированного программирования**

ООП – это новая концепция в технологии программирования по созданию сложных программных комплексов.

ООП основано на трех основополагающих концепциях:

инкапсуляция;

полиморфизм;

наследование.

Замечание: Иногда сюда добавляют еще одну концепцию – концепцию абстракции типов.

**Инкапсуляция** (encapsulation) – это механизм, который объединяет данные и код, манипулирующий с этими данными, а также защищает и то и другое от внешнего вмешательства или неправильного использования.

Когда данные и коды объединены, то говорят, что создается объект, то есть иными словами, объект – это то, что поддерживает инкапсуляцию.

Внутри объекта коды и данные могут быть закрытыми (private), открытыми (public) или защищенными (protected). Закрытые коды или данные доступные только для других частей этого объекта. Если коды и данные являются открытыми, то они доступны и для других частей программы. Использование защищенных данных и кодов занимает промежуточный вариант между использованием открытых и закрытых (см. далее).

**Полиморфизм** (polymorphism) (от греческого слова polymorphos) – это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач.

Выполнение действий в каждом конкретном случае будет определяться типом данных. Тип данных, который используется при вызове функции, определяет, какая конкретная версия функции действительно выполняется. То есть, в С++ можно использовать одно имя функции для множества различных действий. Это называется перегрузкой функции (function overloading).

В более общем смысле концепцией полиморфизма является идея "один интерфейс, множество методов". То есть полиморфизм позволяет манипулировать объектами различной степени сложности путем создания общего для них стандартного интерфейса для реализации похожих действий.

Элементы полиморфизма присутствовали в других языках и раньше (знак "+").

**Наследование** (inheritance) – это процесс, посредством которого один объект может приобретать свойства другого. Точнее, объект может наследовать основные свойства другого объекта и добавлять к ним свойства, характерные только для него.

Наследование позволяет поддерживать концепцию иерархии классов.

Пример; // полиморфизм

В языке Си нахождение абсолютной величины числа требует трех различных функций

abs() – для целых чисел;

labs() – для длинных чисел;

fabs() – для чисел с плавающей точкой.

В С++ используется одна функция abs().

Полиморфизм в жизни – колесо (колесо, рулевой механизм).

Пример; // наследование (иерархия классов)

комната -> дом-> строение -> конструкция.

В настоящее время глобально существует две версии С++:

Традиционная (с начала 90-х годов двадцатого века);

Стандартная (современная,усовершенствованная).

Примеры отличий:

Изменился стиль оформления заголовков (headers).

Появилась инструкция namespace.

// Программа на С++ в традиционном стиле:

#include <iostream.h>

int main()

{

}

// Программа на С++ в современном стиле:

#include <iostream>

using namespace std;

int main()

{

}

iostream.h в С++ имеет то же самое назначение, что stdio.h в Си.

Заголовки уже не обязательно являются именами файлов, поэтому не надо указывать расширение .h, а только имя заголовка в угловых скобках.

В стандартном С++ к стандартным заголовкам Си просто добавляется префикс с и удаляется расширение .h

Пример:

в Си в С++

math.h cmath

string.h cstring

**Пространство имен**

Эта некая объявляемая область, необходимая для того, чтобы избежать конфликтов имен идентификаторов. Традиционно имена библиотечных функций располагались в глобальном пространстве имен (язык Си). Однако содержание заголовков нового стиля помещается в пространстве имен std.

Для того чтобы пространство имен была видимым, используют инструкцию:

using namespace std;

Это инструкция помещает std в глобальное пространство имен.

Комментарии в С++

/\* \*/ многострочный комментарий;

// – однострочный комментарий.

Консольный ввод и вывод в С++

Хотя функции printf() и scanf() доступны в С++, для ввода и вывода используют операторы, а не функции ввода-вывода.

Оператор ввода: cin>>

Оператор вывода: cout<<

Конец строки: endl

В Си это операторы левого и правого сдвига. В С++ они сохраняют свое первоначальное значение, выполняя при этом еще ввод и вывод.

Пример:

cout<<"Это строка вывода на экран.\n";

cout<<10.99;

т.е., в общем виде

cout<<выражение;

int num;

cin>>num; // & ставить не надо.

Операторы << >> ­ – являются примером перегрузки операторов.

//Первая программа на С++

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

printf("Hello C++\n");

int a, b = 10;

double c;

scanf\_s("%lf", &c);

c = sqrt(c);

cout << "a=";

cin >> a;

cout << "b=" << b << " a=" << a <<" c=" << c << endl;

system("pause");

return 0;

}

***Замечание:***

1. При стандартных настройках вариант <iostream.h> не работает.
2. Возможно использование функций языка Си(printf(), scanf()).

Два двоеточия называются оператором расширения области видимости.

//Первая программа на С++ (без использования using namespace std;)

#include<iostream>

#include<cmath>

int main()

{

printf("Hello C++\n");

int a, b = 10;

double c;

scanf\_s("%lf", &c);

c = sqrt(c);

std::cout << "a=";

std::cin >> a;

std::cout << "b=" << b << " a=" << a <<" c=" << c << std::endl;

system("pause");

return 0;

}

**Классы**

Одним из наиболее важных понятий С++ является понятии класса.

Класс – это механизм для создания объектов.

Класс объявляется с помощью ключевого слова class

Синтаксис объявления класса похож на синтаксис объявления структуры.

class имя\_класса {

закрытые функции и переменные класса

public:

открытые функции и переменные класса

} список\_объектов;

В объявлении класса список\_объектов не обязателен, можно объявлять объекты класса позже, по мере необходимости.

Функции и переменные, объявленные внутри объявления класса, становятся членами (members) этого класса.

По умолчанию все переменные и функции, объявленные в классе, становятся закрытыми. То есть, они доступны только для других членов того же класса.

Для объявления открытых членов класса используется ключевое слово public, за которым следует двоеточие. Все переменные и функции, объявленные после public, доступны как для других членов класса, так и для данных любой другой части программы, в которой находится этот класс.

Отличием между структурой и классом в С++ является то, что члены класса по умолчанию, являются закрытыми, а члены структуры – открытыми.

Пример:

class myclass {

// закрытые члены класса

double a;

public:

void set\_a(double num);

double get\_a();

};

В этом примере переменная a – закрытая переменная; set\_a() и get\_a() – две открытые функции. Прототипы функций объявляются внутри класса (функции – члены класса).

После объявления set\_a() и get\_a() в myclass, они еще должны быть определены. Для определения функции-члена класса надо связать имя класс, частью которого является функция-член, с именем функции.

Это достигается путем написания имени функции вслед за именем класса с двумя двоеточиями. Два двоеточия называются оператором расширения области видимости.

void myclass::set\_a(double num)

{

a=num;

}

double myclass::get\_a()

{

return a;

}

Основная форма определения функции-члена:

тип\_возвр\_значения имя\_класса::имя\_функции(список параметров)

{

… // тело функции

}

Создание объектов типа myclass:

myclass ob1; // объект типа myclass

Обращение к членам класса производится через оператор точка (.):

ob1.set\_a(99.9);

Обращение к закрытой переменной a, только через открытую функцию.

// Вторая программа на С++

#include<iostream>

using namespace std;

class myclass {

// закрытые члены класса

double a;

public:

void set\_a(double num);

double get\_a();

};

void myclass::set\_a(double num)

{

a = num;

}

double myclass::get\_a()

{

return a;

}

int main()

{

double b;

myclass ob1; // объект типа myclass

// Обращение к членам класса производится через оператор точка(.) :

ob1.set\_a(99.9);

// Ошибка ( a - закрытая переменная)

// cout << " a=" << ob1.a << endl;

b = ob1.get\_a();

cout << "a=" << b << endl;

system("pause");

return 0;

}