ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧ РАДИОТЕХНИКИ И СВЯЗИ. ЛЕКЦИЯ 1

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Процедурное
программирование.

В общем случае компьютерные языки имеют дело с двумя концепциями — данные и алгоритмы. *Данные* — это информация, которую использует и обрабатывает программа. *Алгоритмы* — это методы, используемые программой. Язык программирования называют процедурным, когда в при использовании акцент ставится на обработку данных с помощью алгоритмов. Суть процедурного программирования заключается в том, чтобы запланировать действия, которые должен предпринять компьютер, и с помощью языка программирования их реализовать. В программе предварительно описывается некоторое количество процедур, которые должен будет выполнить компьютер, чтобы получить определенный результат.

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Процедурное
программиров

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Структурное и модульное программирование.

Для ограничения ветвления внутри кода и упрощения модификации программы используется подход, называемый *структурным программированием.* В языке C++ для реализации такого подхода используются циклы *for, while, do while* и оператор *if else.*

Еще одним важным подходом является *нисходящее программирование*. Главная особенность такого подхода – разделение программы на небольшие, поддающиеся управлению задачи. Такой подход позволяет разрабатывать модули, называемые *функциями*, которые отвечают за выполнение конкретной задачи. В структурном программировании отображено процедурное программирование, программа представляется в виде действий, которые она должна выполнить.

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Структурное и модульное программирование.

```
void hello();

…

void hello()   // определяем функцию hello ()
{
   using namespace std;

   cout << “Hello, World"  << endl;
}              //функции void не имеет возвращаемых значений
```

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Объектно-ориентированное программирование.

Несмотря на то что принципы структурного программирования делают сопровождение программ более ясным, надежным и простым, написать большую программу все еще было непросто. Новый подход к решению этой проблемы был воплощен в объектно-ориентированном программировании (ООП). В отличие от процедурного программирования, в котором акцент делается на алгоритмах, в ООП во главу угла поставлены данные. Вместо того чтобы пытаться решать задачу, приспосабливая ее к процедурному подходу в языке программирования, в ООП язык программирования приспосабливается к решению задачи. Суть заключается в том, чтобы создать такие формы данных, которые могли бы выразить важные характеристики решаемой задачи.

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Объектно-ориентированное программирование.

В языке программирования C++ существуют понятия *класса*, который представляет собой спецификацию, описывающую такую новую форму данных, и *объекта*, который представляет собой индивидуальную структуру данных, созданную в соответствии с этой спецификацией. Например, класс может описывать общие свойства руководителя компании (фамилия, занимаемая должность, годовой доход, необычные способности и т.п.), а объект может представлять конкретного человека. В общем, класс описывает, какие данные используются для отображения объекта и какие операции могут быть выполнены над этими данными.

Подход к проектированию программ, применяемый в ООП, заключается в том, чтобы сначала спроектировать классы, в точности представляющие вес элементы, с которыми будет работать программа.

Переход от объектов класса к проектированию программы называется *восходящим* программированием.

ФИЛОСОФИЯ ПРОГРАММИРОВАНИЯ

Объектно-ориентированное программирование.

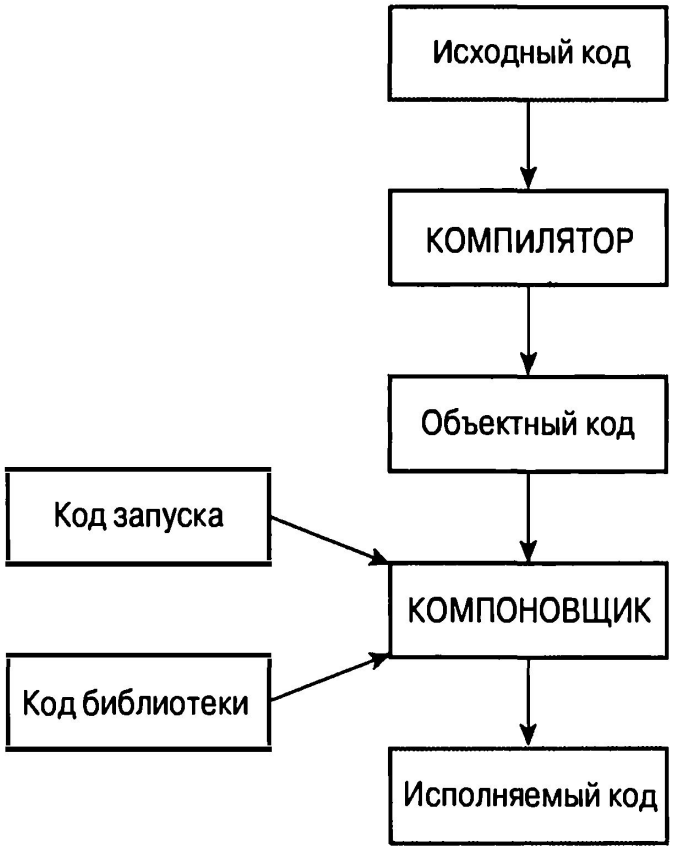
Основные особенности объектно-ориентированного программирования:

1. Наследование;
2. Инкапсуляция;
3. Полиморфизм.

СОЗДАНИЕ ПРОГРАММЫ

Действия, которые должный быть предприняты для написания программы:

1. Создание *исходного кода* программы;
2. Компиляция исходного кода. Для этого необходима среда разработки. При этом создается *объектный код;*
3. Подключение *библиотек.* Библиотеки содержат *функции* для решения определенных задач. В результате получается *исполняемый код.*

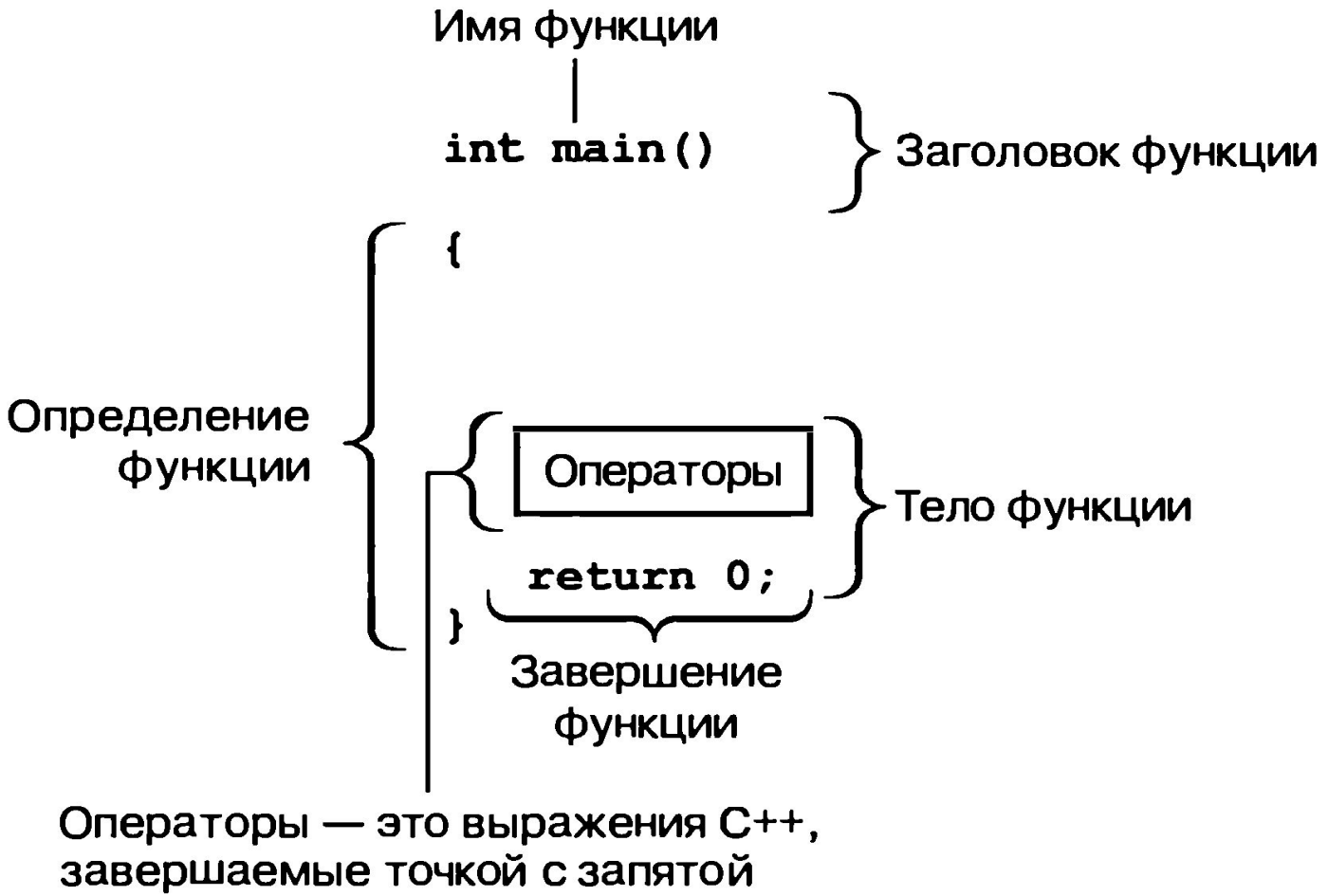


СОЗДАНИЕ ПРОГРАММЫ

Некоторые основные элементы программного кода.

- комментарии, обозначаемые //;
- директива препроцессора #include;
- функция int main();
- директива namespace std;
- тело функции, ограниченное фигурными скобками { и };
- Оператор возврата для прекращения выполнения;
- Функции ввода и вывода cin и cout.

СОЗДАНИЕ ПРОГРАММЫ



СОЗДАНИЕ ПРОГРАММЫ

Функция
main.

В этих строках говорится о том, что существует функция по имени main (), и они описывают ее поведение. Вместе эти строки образуют *определение* функции. Это определение состоит из двух частей: первой строки, int main (), которая называется заголовком функции, и частью, заключенной в скобки ({ и }), которая называется *телом* функции. В заголовке функции вкратце описан ее интерфейс с остальной частью программы, а в *теле* функции содержатся компьютерные инструкции о том, что функция должна делать. В языке C++ каждая полная инструкция называется оператором. Каждый оператор должен завершаться точкой с запятой.

СОЗДАНИЕ ПРОГРАММЫ

Функция
main.

Функция в языке C++ активизируется, или *вызывается*, другой функцией, и заголовок функции описывает интерфейс между функцией и вызывающей ее функцией. Та часть, которая предшествует имени функции, называется *возвращаемым типом функции*; в ней описывается информационный поток, поступающий из функции обратно к функции, которая ее вызвала. Та часть, которая заключена в скобки после имени функции, называется *списком аргументов* или *списком параметров*; в ней описывается информационный поток, который поступает из вызывающей функции к вызываемой функции. Применительно к main() это описание будет немного отличаться, поскольку она, как правило, не вызывается из других частей программы.

Обычно к функции main () обращается код запуска, добавляемый компилятором в вашу программу — он нужен в качестве связующего звена между программой и ОС.

СОЗДАНИЕ ПРОГРАММЫ

Функция
main.

Заголовок

int main ()

говорит о том, что функция main () возвращает целочисленное значение в вызывающую ее функцию, и что функция main () не принимает никакой информации от вызывающей ее функции.

Некоторые программисты используют следующий заголовок и опускают оператор возврата:

void main()

Это логически имеет смысл, поскольку возвращаемый тип void означает, что функция не возвращает значения. Однако хотя данный вариант работает в некоторых системах, он не является частью стандарта C++. Таким образом, в других системах он может дать сбой.

СОЗДАНИЕ ПРОГРАММЫ

Функция
main.

Выполнение программы на C++ всегда начинается с функции main(). Таким образом, если в программе функция main () отсутствует, то такая программа рассматривается как неполная и компилятор выдаст сообщение об отсутствии определения функции main ().

Существует ряд исключений. Например, при программировании для Windows можно написать модуль динамически подключаемой библиотеки (Dynamic Link Library — DLL). Эта библиотека содержит код, который могут использовать другие Windows-программы. Поскольку модуль DLL не является автономной программой, для него функция main () не нужна. Программы, предназначенные для работы в специальных средах, таких как контроллер в автоматическом устройстве, могут не нуждаться в функции main (). Некоторые среды для программирования предоставляют скелетную программу, вызывающую нестандартную функцию наподобие _tmain (); в этом случае имеется скрытая функция main (), которая вызывает _tmain ().

СОЗДАНИЕ ПРОГРАММЫ

Препроцессор и
iostream.

Если ваша программа предназначена для использования обычных средств ввода и вывода в C++, вы предоставляете следующие две строки:

```
#include <iostream>
using namespace std;
```

В языке C++, как и в C, используется *препроцессор*. Препроцессор — это программа, которая выполняет обработку файла исходного кода перед началом собственно компиляции. Препроцессор запускается автоматически во время компиляции программы.

Часть io в iostream означает *input* (ввод) — входные данные программы, и *output* (вывод) — информация, передаваемая из программы. Схема ввода-вывода в C++ включает несколько определений, которые можно найти в файле iostream.

СОЗДАНИЕ ПРОГРАММЫ

Пространство
имен.

Поддержка пространства имен — это средство C++, предназначенное для упрощения разработки крупных программ и программ, в которых комбинируется существующий код от нескольких поставщиков, а также для помощи в организации таких программ. Одна из потенциальных проблем заключается в том, что вы можете работать с двумя готовыми продуктами, в каждом из которых присутствует, функция с одним и тем же именем. В этом случает для того, чтобы получить доступ к нужной функции, нужно перед ее вызовом указать, из какого именно пространства имен будет вызвана функция:

```
std::cout<<“Hello, World”<<std::endl;
```

СОЗДАНИЕ ПРОГРАММЫ

Операторы. Оператор вывода.

```
#include <iostream>

int main()
{
   using namespace std;

   int carrots;              // объявление переменной целочисленного типа

   carrots = 25;             // присвоение переменной значения
   cout << "I have ";
   cout << carrots;          // вывод значения переменной
   cout << " carrots.";
   cout << endl;
   carrots = carrots - 1;    // изменения значения переменной
   cout << "Crunch, crunch. Now I have " << carrots << " carrots." << endl;
   // cin.get();
return 0;
}
```

СОЗДАНИЕ ПРОГРАММЫ

Операторы. Оператор вывода.

Часть, заключенная в двойные кавычки — это сообщение, которое необходимо вывести на экран. В C++ любая последовательность символов, заключенных в двойные кавычки, называется символьной строкой, по-видимому, из-за того, что она состоит из множества символов, собранных в одну большую конструкцию. Запись « означает, что оператор отправляет строку в cout; символы указывают на направление передачи информации. Это предопределенный объект, который знает о том, как отображать разнообразные элементы, включая строки, цифры и индивидуальные символы.

СОЗДАНИЕ ПРОГРАММЫ

Операторы. Оператор ввода.

```
#include <iostream>
#include <cmath>
Int summ(int,int);
int main()
{
   using namespace std;
   const int Months = 12;
int a=1, b=2,c;
c=summ(a,b);
;
// cin.get();
// cin.get();
   return 0;
}

int summ(int a, int b)
{
return a+b;
}
```

СОЗДАНИЕ ПРОГРАММЫ

Работа с данными. Переменные. Типы
данных.

Целочисленные типы: short, int, long, long long, char.

Эти типы позволяют хранить числа со знаками. Для того, чтобы хранить числа
беззнака, необходимо добавить слово unsigned при инициализации переменной.

Пример: (unsigned) int I;

Переменная типа char позволяет хранить символьные значения.

Логический тип: bool.

Может принимать значения true или false. Начальное значение для такой переменной может быть задано неявно, с помощью числа. 0 будет преобразован в значений false, любое ненулевое значение – в true.

Квалификатор const.

Переменную, заданную с таким квалификатором, нельзя изменять.

Пример: const int Months = 12;