Visual Studio 2010 . Структура программы (C++).

План занятия

1. Теоретические аспекты программирования на языке C++
2. Синтаксис и программные конструкции Visual C++.
3. Типы данных C++
4. Структура программы

Цели и задачи изучения темы:

В результате изучения темы студент должен

иметь представление:

- о создании программ на языке Visual C++;

знать:

- цели и задачи прикладного программирования;
- этапы создания программ;
- - Структура программы на языке Visual C++

уметь:

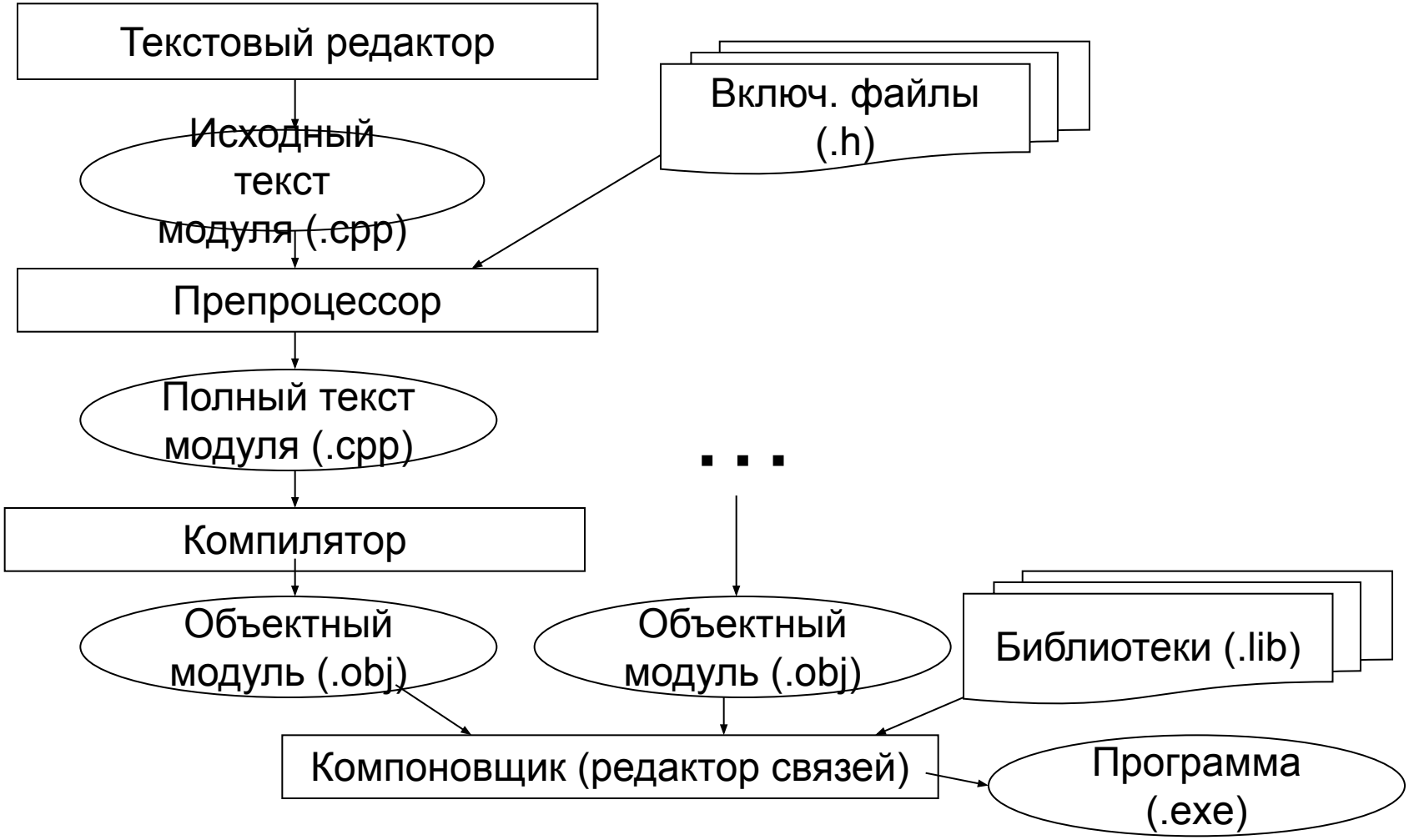
- использовать инструменты прикладного программирования;
- создавать программы на языке Visual C++.

Основные положения

Язык программирования – это формальная знаковая система, которая предназначена для написания программ, понятной для исполнителя (в нашем рассмотрении – это компьютер).

Программа, написанная на языке программирования, состоит из команд (операторов), задающих последовательность действий. Эти действия выполняются над некоторыми объектами. Объектами могут быть числа, текстовые строки, переменные и другие.

Этапы создания программы



Синтаксис и Семантика

- Синтаксис — это правила построения фраз, позволяющие определить, правильно или неправильно написана та или иная фраза. Точнее говоря, синтаксис языка представляет собой набор правил, устанавливающих, какие комбинации символов являются осмысленными предложениями на этом языке.
- Семантика определяет смысловое значение предложений языка. Являясь системой правил истолкования отдельных языковых конструкций, семантика устанавливает, какие последовательности действий описываются теми или иными фразами языка и, в конечном итоге, какой алгоритм определен данным текстом на алгоритмическом языке.

Алфавит языка C++

Алфавит — это фиксированный для данного языка набор основных символов, т.е. "букв алфавита", из которых должен состоять любой текст на этом языке — никакие другие символы в тексте не допускаются.

- Прописные и строчные латинские буквы (различаются в именах), знак подчеркивания
- Цифры (0…9)
- Специальные знаки “ { } , | [ ] ( ) + - * / % \ ; ‘ : ? < = > ! & ~ ^ . #
- Разделители (пробел, табуляция, перевод строки)

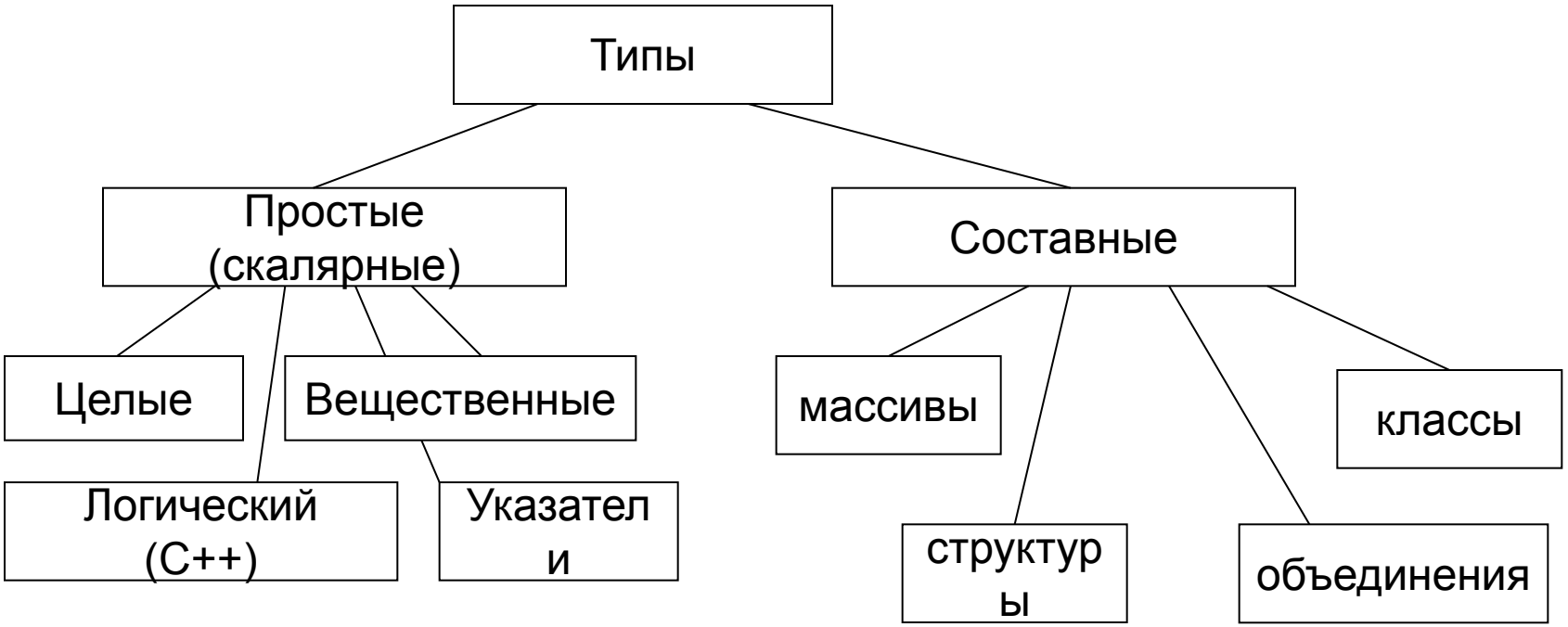
Лексемы C++

- Имена (не рекомендуется начинать с _)
- Ключевые слова
- Знаки операций (одно и двухсимвольные)
- Константы
- Разделители

Комментарии

- однострочные // комментарий
- многострочные /* длинные */

Типы данных C++



Базовые типы

	C/C++	*Pascal (Delphi)*
Целые	char	shortint
	int (short int)	integer
	unsigned char	byte
	unsigned int (short)	word
	long int	longint
	unsigned long int	cardinal
Вещест.	float	single
	double	double
	long double	extended

Специальные типы

- bool – логический (true/false) – в C++

  В C целое значение =0 – ложь,
  не равно 0 - истина
- void – пустой. Используется для обозначения функций без значений и нетипизированных указателей

Тип	**Диапазон значений**	**Размер (байт)**
bool	true и false	1
signed char	-128 … 127	1
unsigned char	0 … 255	1
signed short int	-32 768 … 32 767	2
unsigned short int	0 … 65 535	2
signed long int	-2 147 483 648 … 2 147 483 647	4
unsigned long int	0 … 4 294 967 295	4
float	3.4e-38 … 3.4e+38	4
double	1.7e-308 … 1.7C+308	8
long double	3.4e-4932 … 3.4e+4932	10

Константы

- Целые:
  - *десятичные 123, 0, 98*
  - *восьмеричные 01, 015*
  - *шестнадцатиричные 0xA1, 0X00FF*
- Вещественные 5.8, .2e-3
- Символьные ‘A’, ‘xy’, ‘\n’, ‘\123’, ‘\\’
- Строковые “привет”, “1 \n 2”

Структура программы

<директивы препроцессора>

<функции>

Функция имеет вид

<тип> <имя> (<список параметров>)

{ <операторы>

}

Выполнение начинается с функции main

Пример программы

```
#include <iostream.h>
int main()
{ int a, b; //описание переменных
  cin >> a >> b; //ввод
  cout << “сумма” << a+b; //вывод
  return 0; //возврат
}
```

- Каждая подпрограмма имеет структуру, подобную функции main();
- Каждая программа содержит одну или несколько функций;
- Каждая функция содержит 4 основных элемента:

1. тип возвращаемого значения; Int
2. имя функции; main()
3. список параметров, {return 0;} - заключённый в круглые скобки
4. тело функции

- эта строка значит "*вернуть операционной системе в качестве сигнала об успешном завершении программы значение 0*".

```
#include <iostream>;    //директива процессора, предназначена для включения в
    исходный текст содержимое заголовочного файла, имя которого< iostream>,
    содержащий описания функций стандартной библиотеки ввода/вывода для
    работы с клавиатурой и экраном.
 using namespace std;    //директива означ.что все определённые ниже имена будут
    отн-ся к пространству имён std
 Int  main()     //имя функции,кот.не содержит параметров и должна возвращать
    значение типа Int
{Int a,b;        //объявление двух переменных типа Int - целый тип
 cout <<"введите два целых числа"<<endl;     //оператор вывода данных на экран ,
        << - операция помещения данных в выходной поток;
       endl - манипулятор, переводит сообщение на новую сточку.
 cin >>a >>b;     //оператор ввода данных с клавиатуры,
      >> - операция для извлечения данных из выходного потока, читает значения
    из cin и сохр. их в переменных.
 cout >>"их сумма равна"<<a+b;  //оператор вывода
 return 0;}   //оператор вывода
```

Препроцессор

- Препроцессор — это специальная программа, являющаяся частью компилятора языка Си. Она предназначена для предварительной обработки текста программы.
- Препроцессор позволяет включать в текст программы файлы и вводить макроопределения.
  Работа препроцессора осуществляется с помощью специальных директив (указаний).
- Они отмечаются знаком решетка #. По окончании строк, обозначающих директивы в языке Си, точку с запятой можно не ставить.

Основные директивы препроцессора

#include — вставляет текст из указанного файла
#define — задаёт макроопределение (макрос) или символическую константу
#undef — отменяет предыдущее определение
#if — осуществляет условную компиляцию при истинности константного выражения
#ifdef — осуществляет условную компиляцию при определённости символической константы

#ifndef — осуществляет условную компиляцию при неопределённости символической константы
#else — ветка условной компиляции при ложности выражения
#elif — ветка условной компиляции, образуемая слиянием else и if
#endif — конец ветки условной компиляции
#line — препроцессор изменяет номер текущей строки и имя компилируемого файла
#error — выдача диагностического сообщения
#pragma — действие, зависящее от конкретной реализации компилятора.

Пространство имен

- Каждое имя, определенное в коде, появляющемся внутри пространства имен, включает в себя имя этого пространства имен.
- Все средства стандартной библиотеки ISO/ANSI C++ определены внутри пространства имен по имени std, поэтому каждый элемент стандартной библиотеки, к которому вы можете обратиться в своей программе, имеет свое собственное имя плюс наименование пространства имен — std — в качестве квалификатора.
- Применение полных имен в программе делает код несколько громоздким, поэтому было бы неплохо использовать их простые имена без квалификатора — имени пространства имен std. Две строки программы, которые следуют за директивой #include <iostream>, обеспечивают упомянутую возможность:

- ✔ **using std::cout;**
- ✔ **using std::endl;**
- ✔ **using namespace std**

Функция **main**

Специальная функция **main** - это начальная точка выполнения для всех C и C++ программ (точка входа программы). При создании кода, который соответствует модели программирования Юникод, можно использовать функцию **wmain**, представляющую собой версию функции **main** для расширенных символов.

Проблема русского языка

Проблема русского языка в консольных приложениях заключается в том, что консоль и редактор кода Microsoft Visual Studio поддерживают разные кодовые страницы.

Для того, чтобы увидеть русские символы в консоли необходимо поменять кодовую страницу в консоли, чтобы она соответствовала кодовой странице редактора (1251):

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  system("chcp 1251");
  system("cls");
...
}
```

```
int main()
{
  setlocale("LC_CTYPE", "Russian")
 ...
}
```

Переменные

Формат описания переменных:

[<класс памяти>]<тип><имя>[=<выражение> | (<выражение>)];

Пример:

```
int I,j;
double x;
```

Значение переменных должно быть определено с помощью:

1. оператора присваивания: `int a;` *//описание переменной*
   `int= a;` *//опред.значения. переменной*

2. оператора ввода: `int a;` *//описание переменной*
   `cin>>a;` *//опред.знач.переменной*

3. инициализация – опред.значения переменной на этом этапе описания.
   `int i=100` *//инициализация копией*

Домашнее задание

- Установить среду программирования Visual Studio 2010 на домашних компьютерах.
- Создать проект и произвести компиляцию примера приведенного на занятие:

```
#include <iostream.h>
int main()
{ int a, b; //описание переменных
  cin >> a >> b; //ввод
  cout << “сумма” << a+b; //вывод
  return 0; //возврат
}
```