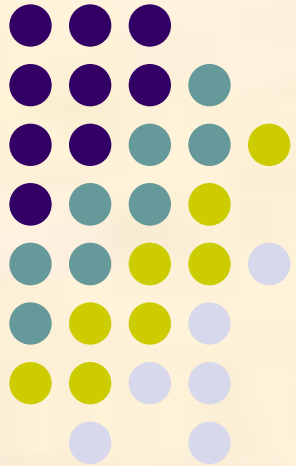
# Лекция 2

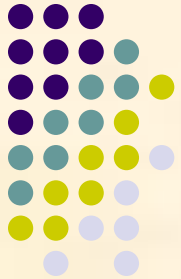
Синтаксис языка C++



# Алфавит языка

- В качестве алфавита языка используются:
- - символы латинского алфавита a, ..., z, A, ..., Z;
- - цифры 0, ..., 9;
- - знаки математических операций + - * / ;
- - специальные символы , . ~ ! # $ % ^ & ( ) _ = | ? : ;" { } [ ]
- Комментарий к тексту программы помещается между символами /* и */ , что исключает сам текст комментария из транслируемой программы.
- Кроме этого, комментарий может помещаться после символов // , исключающих весь последующий до конца строки текст.
- При формировании текстовых строк и для записи комментария к программам допускается использовать символы русского алфавита.

# Идентификаторы и константы

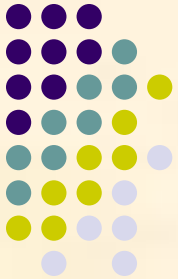


- Идентификатор - "имя" переменной, функции или типа данных, определяемых программистом.
- В качестве идентификатора может использоваться последовательность, не содержащая пробелов, из символов латинского алфавита, цифр и символа _ .
- Идентификатор не должен начинаться с цифры и может содержать до 32 символов.
- В отличие от Паскаля, в C и C++ следует различать строчные и прописные символы при записи идентификаторов. Так, например, **time, tiMe** и **Time** - это разные имена.

# Константы

- Константы целого типа могут задаваться в восьмеричной, десятичной и шестнадцатиричной системах счисления. Для записи констант целого типа используются символы 0, ..., 9, a, b, c, d, e, f, A, B, C, D, E, F.
- Константы вещественного типа занимают в памяти 8 байт (64 бита) и состоят из целой части (со знаком); десятичной точки, отделяющей дробную часть от целой; символа экспоненты e или E: **1.75**, **2.5e-2**.
- Символьная константа представляется символами кода ASCII, заключенными в апострофы: **'A', 'b', '8', '#',** и т.д.
- Строковые константы представляются последовательностью символов кода ASCII, заключенной в кавычки: **"Happy Birthday To You!".**
- В строковых и символьных константах возможно добавление специальных служебных символов, например: **'\n'** переводит строку,

# Типы данных



- В языках С и С++ есть четыре основных типов данных:
- Символьные переменные – **char** - занимают в памяти 1 байт, что позволяет сохранять 256 целых чисел в диапазоне от -128 до +127. Это позволяет в необходимых случаях отождествлять каждое представимое число с кодом соответствующего символа таблицы.
- Целые переменные – **int** - целые числа. Назначение переменных этого типа - счетчики циклов, целые числа небольшого диапазона. Размер диапазона для числовых типов зависит от платформы, под которую написан компилятор.
- Вещественные переменные – **float** – предназначены для представления вещественных чисел в экспоненциальной форме: **±0.XXXXXXX e±YY.**
- Вещественные переменные двойной точности – **double.**
- Перечисленные типы имеют несколько дополнительных, редко используемых модификаций.

# Структура программы

- Структура программы на C/C++ схожа со структурой программы на Паскале, однако имеет целый ряд важных особенностей.
- Файлы, содержащие тексты программ на языке C как правило имеют расширение *.c, содержащие тексты программ на C++ имеют расширение *.cpp.
- Структура программы:
- ***препроцессор***
- ***внешние типы, переменные, процедуры и функции***
- ***заголовок основной функции***
- **{**
- ***тело основной функции***
- **}**

# Препроцессор

- В современном языке C++ основная задача препроцессора – обеспечить подключение необходимых внешних библиотек (модулей).
- Подключение дополнительного модуля производится при помощи команды:
- **#include <файл заголовка>**
- Под файлом заголовка модуля понимают специальный файл вида ***.h** в котором дается интерфейсная часть подключаемого модуля.
- Для работы программы с использованием языка C++ необходимо подключать заголовки **stdlib.h** и **iostream.h**, для работы процедур ввода-вывода языка C надо подключить заголовок **stdio.h**. Для подключения математической бибиотеки используется заголовок **math.h**.
- В самых новых версиях языка C++ **.h** после названия заголовка писать необязательно.

# Описание переменных

- Для описания переменных используются команды следующего вида:
- ***Тип переменной*** *список переменных;*
- Например:
- **float a,x,y;** //Описаны вещественные переменные a,x,y.
- **int i,j;** //Описаны целые переменные i,j.
- В отличие от языка Паскаль в C++ описания локальных переменных и типов не выносятся за пределы тела функции. Переменные могут определяться в тексте программы по мере надобности. Данные переменные действуют от момента описания до конца функции.
- Внешними называются переменные, определяемые вне всех функций. Данные переменные действуют от момента описания до конца программы.
- В процесс описания переменной можно задать ее начальное значение:
- **float a=2;**

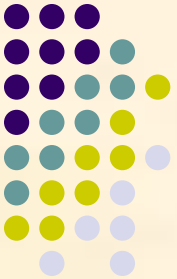
# Описание функций

- Для описания функций используются команды следующего вида:
- ***тип_возвращаемого_значения*** *имя_функции ( список_аргументов )*
- *{ тело_функции;*
- *return выражение;*
- *}*
- В качестве типа_возвращаемого_значения можно указывать ключевое слово **void**, обозначающее, что функция не возвращает никакого значения в вызвавшую ее программу (аналог **процедур** в Паскале).
- Пример, функция возведения вещественного числа в куб:
- **float qub(float x)**
- **{**
- **return x*x*x;**
- **}**

# Основная функция

- Основной функцией программы в C++ называют функцию, выполняющуюся при запуске программы.
- Основная функция имеет имя **main**. Как правило этой функции не нужны аргументы и выводимый тип.
- **void main() {}**
- Иногда среды разработки предлагают в начале работы собственный вид данной функции.
- Таким образом, рекомендуемая заготовка для написания программы на языке C++ имеет следующий вид:
- **#include <iostream.h>**
- **#include <stdlib.h>**
- **void main()**
- **{**
- **// текст программы**
- **cin.get(); // приостановка выполнения**
- **}**

# Пример программы на C++



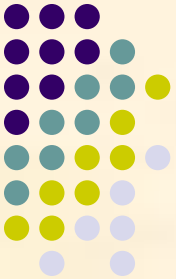
- Рассмотрим пример простейшей программы на языке C++.
- Данная программа вычисляет значение некоторой функции действительного числа.
- В программе используются типичные для языка C++ синтаксические конструкции.

```
#include <iostream.h>
#include <stdlib.h>
#include <math.h>

float f(float x)
{
  return sin(x);
}

void main()
{
     float a,x=2;
     a=f(x);
     cout<<"Sinus"<<'\n'<<a;
     cin.get();
}
```

# Операции языка C++

- Основные арифметические операции: **+, -, *, /.**
- В случае деления целых чисел результат будет частным двух чисел. Для получения остатка используется операция **%.**
- Для сокращения записи вместо операций типа **a=a+b** применяют также запись **a+=b**. Аналогичный смысл имеют записи: **a-=b, a*=b, a/=b, a%=b.**
- Две операции ++ и - - введены для увеличения (уменьшения) переменных ровно на 1, что часто встречается при циклических вычислениях, индексации массивов и в ряде других случаев.
- a++; // эквивалент операции a = a + 1 ; или a + = 1 ;
- a- - ; // эквивалент операции a = a - 1 ; или a - = 1 ;
- Если символы ++ (--) расположены слева от переменной, то сначала переменная увеличивается (уменьшается) на 1, а затем производится вычисление выражения.
- И наоборот, если символы сложения (вычитания) расположены справа от идентификатора, первоначально вычисляется арифметическое выражение, а затем переменная увеличивается (уменьшается) на 1.

# Операции языка C++



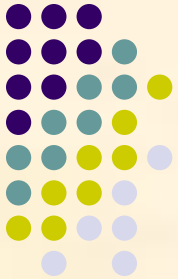
- Пример:

```
main()
{  int n, b, c;
n=1;
b=10*(n++);//  сначала вычисляется b = 10 * 1 = 10, а затем n = n + 1 = 2;
c=10*(++n);//  сначала вычисляется n = n + 1 = 3, а после этого c = 10 * 3 = 30;
}
```

- Логические операции:
- В C++ в качестве FALSE используется значение целого типа (int), равное 0, в качестве TRUE - любое ненулевое значение целого типа.

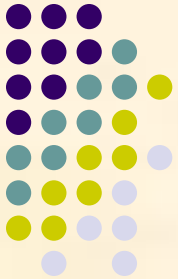
| | | | |
|---|---|---|---|
| == | равно | <,>,<=,>= | Больше,меньше |
| != | не равно | && | логическое И (and) |
| ! | отрицание (not) | \|\| | логическое ИЛИ (or) |

# Операторы языка C++



- **1. Оператор (или операция) присваивания =.**
- Примеры:
- **a=b=2;** //присваивание числового значения
- **b=a+c**; // присваивание результата действия
- **2. Составной оператор** – последовательность любых операторов, заключенная в фигурные скобки **{ }**, которые являются в С и С++ аналогом слов **begin** и **end**.
- **3. Условные операторы**
- **Логический оператор присваивания:**
- **переменная =(логическое_выражение)? выражение1:выражение2;**
- выполняется следующим образом: если логическое_выражение истинно, то переменной присваивается результат выражения1, а в противном случае - результат выражения2.
- **main()**
- **{float a, b=0.5, d=25.0e10;**
- **a=(b>0)? d*1.0e-12: 3.1415926;**
- **// если b больше 0, то a = d * 1.0e-12 = = 0.25,**
- **// в противном случае a = 3.1415926.**
- **}**

# Операторы языка C++

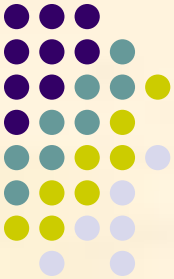


- **Условный оператор if:**
- **if(выражение ) оператор;**
- Если выполняется логическое выражение, то будет выполняться оператор.
- Пример:
- **if (f==0 || f==1) { g=1; h=2; i=3; }**
- Другая форма оператора **if – оператор ветвления**:
- **if(выражение) оператор_1; else оператор_2;**
- Если истинно выражение, то выполняется оператор_1, иначе - оператор_2.
- Оператор переключения **switch**:
- **switch(выражение)**
- **{case const_1: оператор_1; break;**
- **case const_2: оператор_2; break;**
- **// . . .**
- **default:оператор_N; break;**
- **}**
- выражение - целочисленное арифметическое выражение;
- const_i - символьная или целая константа;
- break - оператор прерывания, обеспечивающий выход из оператра switch.

# Операторы языка C++

- **4. Операторы цикла**
- **Оператор цикла for:**
- **for (инициализация; условие; изменение) оператор;**
- **Инициализация** - определение начальных значений меняющихся при выполнении цикла величин.
- **Оператор** - повторяющиеся операции, на которые распространяется оператор цикла; в случае, когда необходимо циклическое выполнение группы операторов, их следует заключать в операторные скобки.
- **Изменение** - изменение переменных величин по окончании очередного цикла.
- **Условие** - логическое выражение, определяющее условие выполнения цикла.
- Поля **инициализация** и **изменение** могут содержать список операций, перечисляемых через запятую. Следует отметить, что любое из указанных полей может отсутствовать; при этом разделители ; должны сохраняться.

# Операторы языка C++



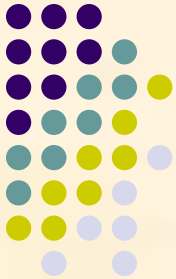
- Пример: Программа вычисления суммы первых ста натуральных чисел.

```
#include <iostream.h>
void main()
{    int sum, count;
     for(count=1, sum=0; count<=100; count++) sum+=count;
     cout<<"Summa ravna "<<sum; //вывод на экран
     cin.get();
}
```

- **Оператор цикла while:**
- **while (выражение) оператор;**
- Пока справедливо логическое **выражение**, будет выполняться **оператор.**

```
#include <iostream.h>
void main()
{
   int sum=0, count=1;
   while (count<=100) sum+=count++;
   cout<<"Summa ravna "<<sum;
   cin.get();
}
```

# Операторы языка C++

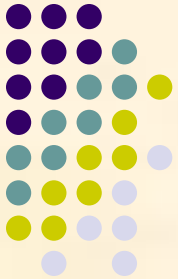


- **Оператор цикла do_while:**
- **do оператор; while(выражение);**
- **Оператор** будет выполняться, пока справедливо логическое **выражение.**
- Пример:
- **#include <iostream.h>**
- **void main()**
- **{**
- **int sum=0, count=1;**
- **do sum+=count++; while (count<=100);**
- **cout<<"Summa ravna "<<sum;**
- **cin.get();**
- **}**
- В отличие от языка Паскаль оператор цикла с постусловием задает условие продолжения, а не окончания цикла.
- Как можно увидеть из приведенных примеров, программы на C++ отличаются высокой компактностью, однако чтение текстов подобных программ и их анализ достаточно сложен.

# Ввод-вывод данных

- В языке С существуют достаточно сложные процедуры **printf** и **scanf** для вывода и ввода данных на экран. Данные процедуры содержатся в модуле **stdio**.
- В языке С++ появились более простые и удобные функции ввода-вывода **cin** и **cout**, находящиеся в модуле **iostream**.
- Для вывода данных на экран используется функция **cout<<:**
- Настройка вывода осуществляется при помощи команд:
- **cout.width(n),** где n – ширина поля вывода и
- **cout.precision(m),** где m – точность вывода вещественных чисел.
- Пример:
- **#include <iostream.h>**
- **void main()**
- **{ float pi=3.1415926;**
- **cout.precision(5);**
- **cout<<"Chislo pi:" << '\n' <<pi;**
- **cin.get();**
- **}**
- **Результат: Chislo pi:**
- **3.1416**

# Ввод-вывод данных



- Для ввода данных с клавиатуры используется процедура **cin>>.**
- Пример: решение квадратного уравнения

```
#include <iostream.h>
void main()
{ float a,b,c,d,x1,x2;
  cout<<"a*x*x+b*x+c=0"<<'\n'<<"Vvedite a:";
  cin>>a;
  cout<<"Vvedite b:"; cin>>b;
  cout<<"Vvedite c:"; cin>>c;
  d=b*b-4*a*c;
  if (d<0) cout<<"Net korney";
  else {
       x1=(-b+sqrt(d))/(2*a);
       x2=(-b-sqrt(d))/(2*a);
       cout<<"x1="<<x1<<'\n'<<"x2="<<x2;
      }
  cin.get();
}
```