ГУАП

guap.ru

Информатика. Информационные технологии

План лекции

- Язык программирования C++
  - Типы данных в C++
  - Арифметические, логические и другие операторы.
  - Данные в C++ (переменные и массивы)
  - функции C++
  - циклы for, do-while, while-do
  - условные операторы if else
  - переключатель switch
  - компилирование программ в C++

Типы данных в C++

- В C++ типы данных определяют способ хранения информации в памяти.
- Перед использованием переменных в C++ необходимо объявить их тип.

Тип данных	Размер в байтах	Диапазон допустимых значений
char	1	от -128 до 127
unsigned char	1	от 0 до 255
short	2	от -32 768 до 32 767
unsigned short	2	от 0 до 65 535
long	4	от -2 147 483 648 до 2 147 483 647
unsigned long	4	от 0 до 4 294 967 295
int	4	совпадает с long
unsigned int	4	совпадает с unsigned long
float	4	от 1.2E-38 до 3.4E38
double	8	от 2.2E-308 до 1.8E308
bool	1	true или false

Оператор	Описание	Пример
Арифметические операторы		
+	сложение	x = x + z;
-	вычитание	x = x - z;
*	умножение	x = x * z;
/	деление	x = x / z;
Операторы присвоения		
=		
+=; -=; *=; /=;	Сложение (вычитание, умножение, деление) с присвоением	x += 10; (то же, что и x = x+10)
&=	Поразрядное И с присвоением	x &= 10;
|=	Поразрядное ИЛИ с присвоением	x |= 10;
Логические операторы		
&&	Логическое И	if (x && 0xFF) {…}
||	Логическое ИЛИ	if (x || 0xFF) {…}

Оператор	Описание	Пример
Операторы отношения		
==	Равно	if (x == 10) {…}
!=	Не равно	if (x != 10) {…}
<	Меньше	if (x < 10) {…}
>	Больше	if (x > 10) {…}
<=	Меньше и равно	if (x <= 10) {…}
>=	Больше и равно	if (x >= 10) {…}
Унарные операторы		
*	Косвенная адресация	int x = *y;
&	Взятие адреса	int * x= &y;
~	Поразрядное НЕ	x &= ~0x02;
!	Логическое НЕ	if (!valid) {…}
+	Инкремент	x ++
--	Декремент	x --

Переменные в C++

- Переменная - это имя присвоенное некоторому участку памяти. После объявления переменной ее можно использовать для операций с данными в памяти.
- Переменные, которые объявлены, но не инициализированы, содержат случайные значения

```
int x;          // объявлена переменная 'x' целого типа
x = 100;        // теперь 'x' содержит значение 100
x += 50;        // теперь 'x' содержит значение 150
int y = 150;    // 'y' объявлена и инициализирована значением 150
x += y;         // теперь 'x' содержит значение 300
x++;            // теперь 'x' содержит значение 301
```

Массивы C++

Массив это набор данных. Для хранения данных типа int нужно сделать следующее объявление:

```
int myArray[5]
```

Так как каждое число типа int требует 4 байта, то для данного массива потребуется 20 байт

Массив может быть объявлен и инициализирован одновременно:

```
int myArray[5]={1,2,3,4,5};
```

```
myArray[0]=1
```

```
myArray[0]=2
```

```
myArray[0]=3
```

```
myArray[0]=4
```

```
myArray[0]=5
```

Массивы могут быть многомерными

```
myArray[0][1]=1
```

Символьные массивы

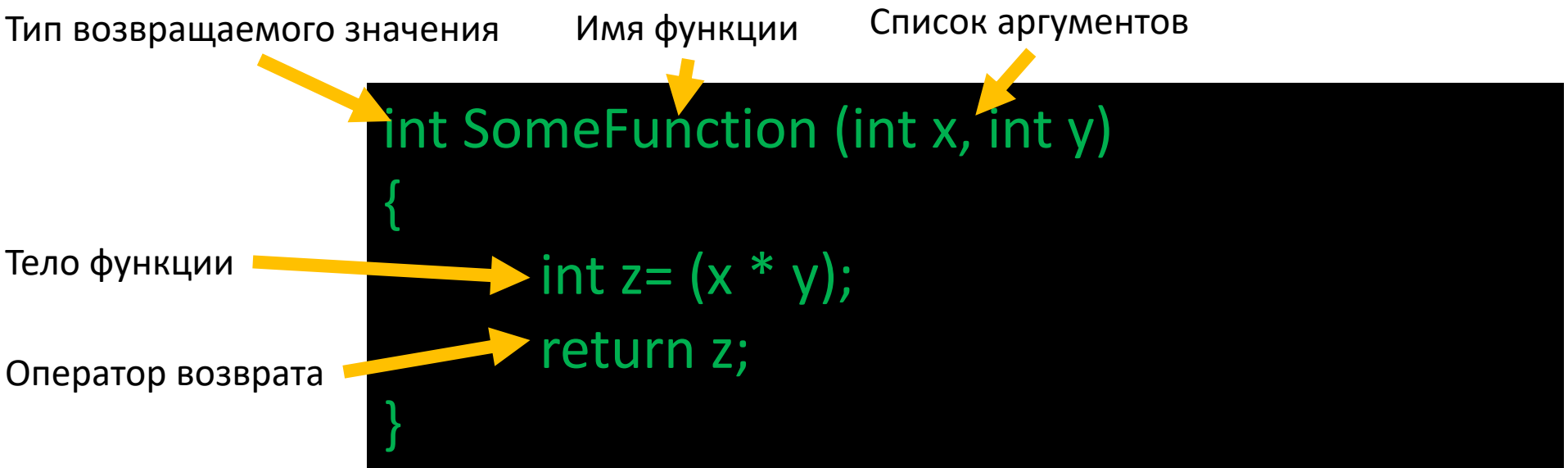
- В C++ отсутствует поддержка строковых переменных (переменных содержащих текст). Для представления строк в программах на C++ используются массивы переменных типа char.

```
char text[]= “This is a string.”;
```

- В этой строке 17 символов, однако под нее выделяется 18 байт, так как любая строка дополняется нуль символом \0. Он интерпретируется как конец строки.

Функции C++

- Функции – блоки кода, не входящие непосредственно в основную программу. Они выполняются/вызываются по мере необходимости.
  - Простейшая функция не принимает никаких аргументов и возвращает число типа void (то есть не возвращает ничего)
  - Другие функции могут принимать несколько аргументов и возвращать некоторое значение.



Основные свойства функций в C++

- Функция может принимать любое количество аргументов или не иметь аргументов вовсе.
- Функция может возвращать значение, но это не обязательно.
- Функция типа void не возвращает никаких значений.
- Если указано, что функция возвращает значение, то она должна содержать оператор return, возвращающей это значение.
- Функция может иметь любое количество аргументов, но возвращает всегда только одно
- Аргументы могут передаваться функциям по значению, через указатели или по ссылке

Объявления функций в C++

- Прототип функции – все создаваемые в программе функции необходимо объявить перед функцией main()
- содержание функций обычно описывается после функции main()

```
#include <iostream>
//functions description
void HelloWorld();
//main function
void main ()
{
                    HelloWorld();
}
void HelloWorld()
{
          std::cout<< “Hello World”<<std::endl;
}
```

Функция main()

- Программа на C++ должна иметь функцию main() - эта функция служит точкой входа в программу
- Функция main() является такой же функцией как и другие и имеет ту же базовую структуру.
- Функцию main() вызывают при запуске программы.

```
int main (int argc, char *argv)
```

```
grep WM_KILLFOCUS -d -i
```

```
argc        содержит 4
argv[0]     содержит c:\bc5\bin\grep.com
argv[1]     содержит WM_KILLFOCUS
argv[2]     содержит -d
argv[3]     содержит -i
```

количество аргументов функции main()
расположение программы

Циклы

- **Цикл** (loop) – это конструкция языка программирования, которая используется для выполнения каких-либо действий определенное количество раз.
- Цикл состоит из:
  - Точка входа
  - Тело цикла (тело цикла содержит описание операций, производимых за один проход)
  - Точки выхода
  - условного выражения, определяющего момент выхода из цикла.
  - Дополнительных операторов:
    - break (немедленное прерывание цикла)
    - continue (переход к началу цикла без выполнения команд, расположенных после данного ключевого слова)

Работа циклов

- на первом этапе поле входа в цикл проверяется условное выражение
- Если выражение верно выполняется тело цикла
- Достигнув точки выхода происходит переход к точке входа
- Если :
  - выражение истинно – цикл повторяется
  - выражение ложно – обрыв цикла

Исключение:

В цикле **do-while** проверка истинности происходит в конце цикла

Цикл for

точка входа

- Оператор цикла **for:**

```
for (initial; condition_expression; adjust) {
        statements;
}
```

точка выхода

тело цикла

- Пример выполнения цикла for:

```
for (int i=0; i<=10; i++){
std::cout << "Hello World "<<i<<std::endl;
}
```

Блок кода, обозначенный как statements выполняется до тех пор, пока условное выражение **condition expression** истинно (**True or >0**)

После выполнения блока кода **statements**, переменная цикла модифицируется оператором **adjust**

```
Hello World 0
Hello World 1
Hello World 2
Hello World 3
Hello World 4
Hello World 5
Hello World 6
Hello World 7
Hello World 8
Hello World 9
Hello World 10
```

Цикл while

Тип данных bool

- Оператор цикла **while:**

```
while (condition_expression) {
        statements;
}
```

Блолк **statements** выполняется до тех пор пока **condition_expression** имеет значение **true** (не равно нулю)

Изменение переменной цикла должно происходить в теле цикла.

Цикл заканчивается в том случае, когда выражение **condition_expression** станет равным **false** (нулю)

- Пример выполнения цикла while:

```
int f0=0;
int f1=1;
while (f<fmax){
        f=f0+f1;
        f0=f1;
        f1=f;
        std::cout << "Fibonacci "<<f<<std::endl;
        }
```

```
Fibonacci 1
Fibonacci 2
Fibonacci 3
Fibonacci 5
Fibonacci 8
Fibonacci 13
Fibonacci 21
Fibonacci 34
Fibonacci 55
Fibonacci 89
Fibonacci 144
```

Цикл do-while

- Оператор цикла **while:**

```
do {
        statements;
} while (condition_expression) ;
```

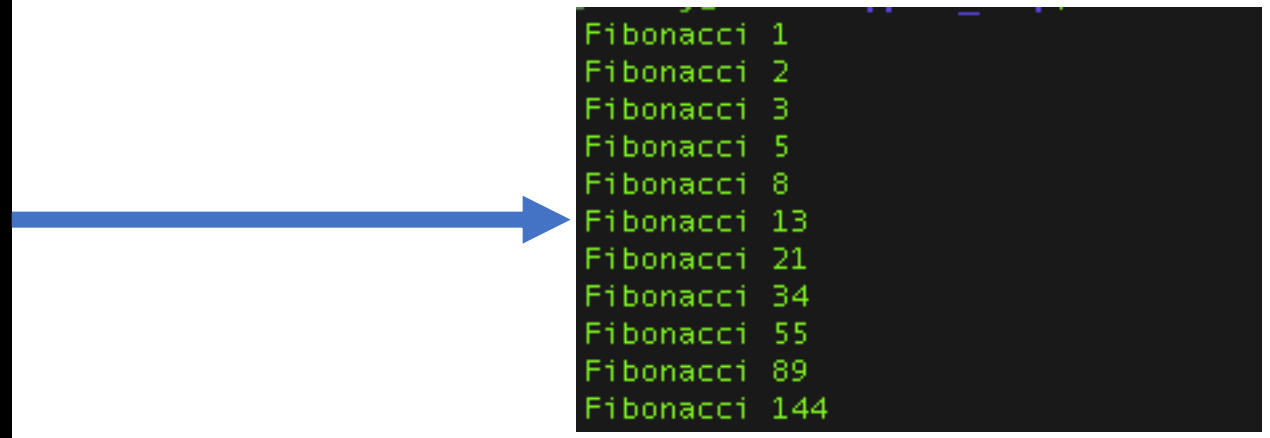
Блолк **statements** выполняется до тех пор пока **condition_expression** имеет значение **true** (не равно нулю)

Изменение переменной цикла должно происходить в теле цикла.

Цикл заканчивается в том случае, когда выражение **condition_expression** станет равным **false** (нулю)

- Пример выполнения цикла do:

```
int f0=0;
int f1=1;
do {
    f=f0+f1;
    f0=f1;
    f1=f;
    std::cout << "Fibonacci "<<f<<std::endl;
    } while (f<fmax);
```



Оператор goto

- Оператор **goto** осуществляет безусловный переход к строке, ранее заданной меткой **label**

```
        goto label;
…..

label:
```

- Пример цикла, построенного с помощью **goto:**

```
bool done – false;
StartPoint:

//some code

if (!done) goto (StartPoint);

// end of the cycle
```

Оператор if

- Оператор if используется для проверки условия и последующего выполнения блока кода в зависимости от того, истинно это условие или ложно

```
if (condition_expression) {
        true_statements;
}
else {
        false_statements;
}
```

Если условное выражение condition_expression истинно, то выполняется блок кода true_statements.

Оператор else необязательный блок кода false_statements выполняется если condition_expression ложно (false)

```
if (condition_expression_1) {
        true_statements_1;
}
else if (condition_expression_2) {
        true_statements_2;

else {
        false_statements;
}
```

Если условное выражение condition_expression_1 истинно, то выполняется блок кода true_statements_1. Если это выражение ложно и истинно выражение condition_expression_2 выполняется блок кода true_statements_2.

Обратите внимание!

```
if (x==10) ;
        DoSomething(x);
```

При такой записи блок кода DoSomething(x) – выполняется в любом случае, так как после оператора if стоят ; которые обозначают окончание оператора

```
if (x==10)
        DoSomething(x);
```

Данная запись корректна, блок кода DoSomething(x) выполняется только если условие x==10 выполнено (True)

Оператор switch

```
switch (value) {
        case 0 : {
                DoSome_1();
                break;
        }
        case 10 : {
                 DoSome_2();
                break;
        }
        case 20 :
        case 30 : {
                DoSome_3();
                break;
        }
        default : {
                DoSome_4();
        }
}
```

- оператор switch выполняет один или несколько блоков кода в зависимости от значения выражения.
- оператор break означает выход из блока switch
- блок кода за default выполняется если совпадений не было
- блок DoSome_3() выполнится если value = 20 или 30

Компилирование программ на C++

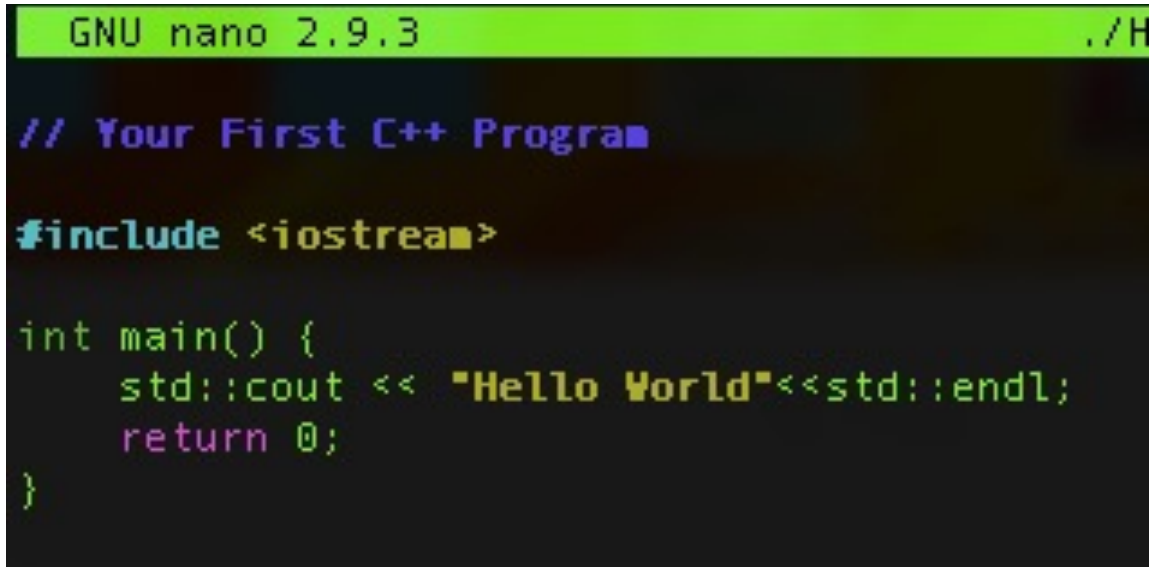
- Компилятор – программа, переводящая текст, написанный на языке программирования в набор машинных кодов
- Зачем нужно компилирование – исходный c++ файл это всего лишь код, его невозможно запускать как программу или использовать как библиотеку

- Состав компилятора:
  - cpp – препроцессор
  - as – ассемблер
  - g++ - сам компилятор
  - ld - линкер

```
pi@raspberrypi:~ $ g++ --version
g++ (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

pi@raspberrypi:~ $
```

C++ Hello world



Строки кода начинающиеся с // являются комментариями и используются для документации программы

Выражение – фрагмент кода, в котором вычисляется некоторое значение

Оператор – это закрытое выражение, для закрытия выражений используется символ ;

Скобки { обозначают начало и конец блока кода, связанных с циклами, функциями, оператором if итд. }

выражение

c = a+b;

оператор

Для использования cout необходимо сообщить компилятору описание класса iostream, это называется объявлением класса iostream. Оно находится в файле IOSTREAM.H этот файл называется заголовочным (header file).

Процесс компиляции программы на C++

1. **Препроцессинг** (использование препроцессора cpp). Преобразование программы для дальнейшего компилирования.

```
cpp -E ./Hello_World.cpp -o ./Hello_World.ii
```

2. **Компиляция** – преобразование полученного на прошлом шаге кода в ассемблерный код. Это промежуточный шаг между высокоуровневым языком и машинным (бинарным кодом). **Ассемблерный код** – доступный для понимания человеком представление машинного кода.

```
g++ -S ./Hello_World.cpp -o ./Hello_World.s
```

3. **Ассемблирование** – процесс перевод ассемблерного кода в машинный код и сохранение его в объектном файле.

```
as ./Hello_World.s -o ./Hello_World.o
```

Объектный файл – созданный ассемблером хранящий кусок машинного кода

4**. Компоновка** (линковка) – сборка всех объектных файлов и статических библиотек в единый исполняемый файл

```
g++ ./Hello_World.o -o ./Hello_World
```