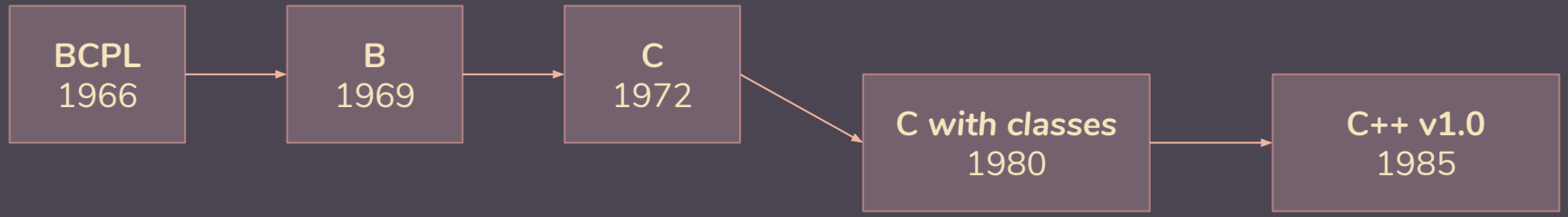
Кирилл Волков @ MERA
github.com/vulko/Cpp_Basics_Lectures

C++

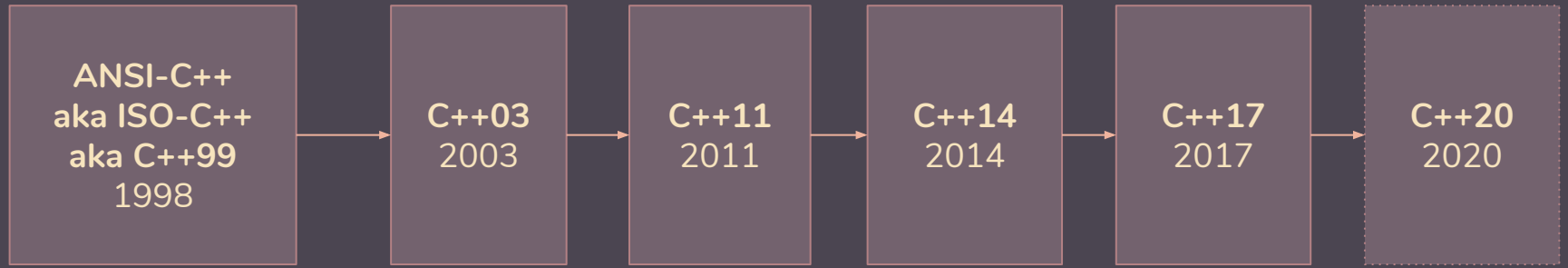
История языка программирования C++.
Основы синтаксиса, структура кода программы.
Модули и пространства имен.
Примитивные типы данных.

История языка

Времена мужиков в лосинах...



Современность...



Структура программы

main.h

```
int main() {
    return 0;
}
```

```
> sudo apt-get update
> sudo apt-get install g++
```

```
> cd location_to_main.cpp
> g++ main.cpp
```

main.cpp

```
#include "main.h"

int main() {
    int retVal = someFunction(true);
    return retVal;
}

int someFunction(bool doSomething) {
    if (doSomething) {
        // do something
        return 0;
    } else {
        return -1;
    }
}
```

#define

```
#define HELLO    "Привет, это пример работы с define"
#define FIVE     5
#define TEN      10
#define FIFTEEN  15
#define SUM(a, b) (a + b)
#define RESULT_MSG println("Сумма = %d", result)

int main() {
    println(HELLO);
    int x = FIVE;
    int y = TEN;
    int result = SUM(x, y);

    if (result != FIFTEEN) {
        println("Кажется что-то пошло не так...");
        return -1;
    }

    RESULT_MSG;
    return 0;
}
```

```
> Привет, это пример работы с define
> Сумма = 15
```

#ifdef #else #endif

```
#define FAIL
#define FIVE     5
#define TEN      10
#define FIFTEEN  15
#define SUM(a, b) (a + b)

int main() {
    int x = FIVE;

#ifdef FAIL
    int y = TEN;
#else
    int y = TEN + 1;
#endif
    int result = SUM(x, y);

    ...
}
```

```
> Кажется что-то пошло не так...
```

#if defined #elif #undef

```
#if defined(WIN32)
#include "win32.h"
#elif defined(WIN64)
#include "win64.h"
#elif defined(LINUX)
#include "linux.h"
#else
#include "default_os.h"
#endif

int main() {
    ...
}
```

```
#if defined(WIN32)
#undef WIN64
#undef LINUX
#include "win32.h"
#elif defined(WIN64)
#undef LINUX
#include "win64.h"
#elif defined(LINUX)
#include "linux.h"
#else
#include "default_os.h"
#endif
```

Заголовочные файлы

```
// hello.h
#ifndef HELLO_H
#define HELLO_H

#include <stdio.h>

void sayHello();

#endif
```

объявление

Подключение другого модуля

```
// main.h
#ifndef MAIN_H
#define MAIN_H

#include <hello.h>

#endif
```

```
// hello.cpp
#include "hello.h"

void sayHello() {
   printf("Hello Mr. Monkey");
}
```

имплементация

```
// main.cpp
#include "main.h"

int main() {
   sayHello();

   return 0;
}
```

имплементация

Пространства имен

```
// hello.h
#ifndef HELLO_H
#define HELLO_H

#include <stdio.h>

void sayHello();
bool init();

#endif
```

Повторное объявление

объявление

```
// main.h
#ifndef MAIN_H
#define MAIN_H

#include <hello.h>
bool init();

#endif
```

```
// hello.cpp
#include "hello.h"

void sayHello() {
   printf("Hello Mr. Monkey");
}
```

```
// main.cpp
#include "main.h"

bool init() { return true; }

int main() {
   sayHello();

   return 0;
}
```

Пространства имен

```
// hello.h
#ifndef HELLO_H
#define HELLO_H

#include <stdio.h>

namespace Hello {
    void sayHello();
    bool init();
}

#endif
```

```
// main.h
#ifndef MAIN_H
#define MAIN_H

#include <hello.h>
bool init();

#endif
```

```
// main.cpp
#include "main.h"

bool init() { return true; }

int main() {
    Hello::sayHello();

    return 0;
}
```

Уже не повторное объявление

Пространство имен

использование

объявление

Пространства имен

namespace MathItems

- Number.h
- SimpleNumber.h
- ComplexNumber.h

namespace MathFunctions

- Function.h
- AddFunction.h
- MultiplyFunction.h

Примитивные типы

Целочисленные знаковые

- `int8_t`
- `int16_t`
- `int32_t`
- `int64_t`

`char`	1B
`short int`	2B
`int`	4B
`long int`	8B

Целочисленные беззнаковые

- `uint8_t`
- `uint16_t`
- `uint32_t`
- `uint64_t`

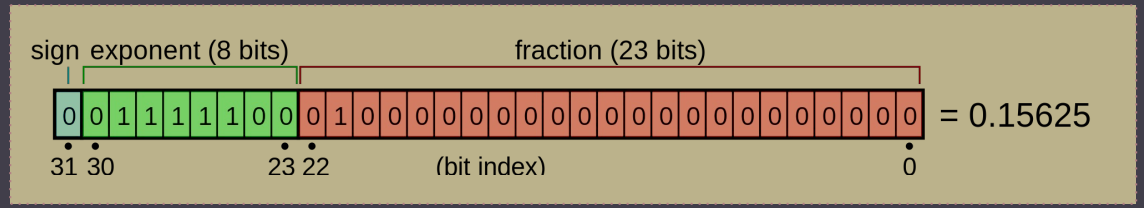
Примитивные типы

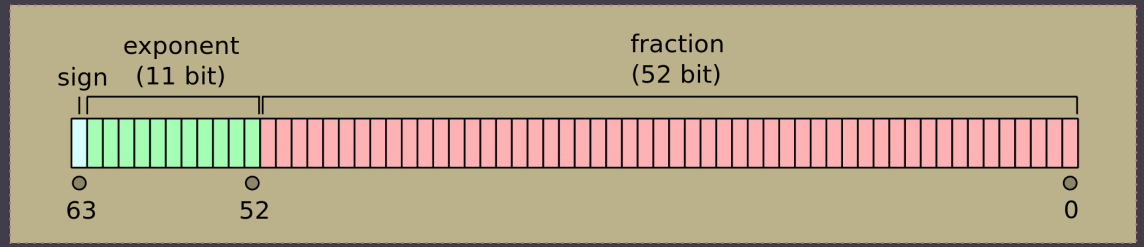
Type specifier	Equivalent type	Width in bits by data model				
		C++ standard	LP32	ILP32	LLP64	LP64
`short`	`short int`	at least **16**	**16**	**16**	**16**	**16**
`short int`						
`signed short`						
`signed short int`						
`unsigned short`	`unsigned short int`					
`unsigned short int`						
`int`	`int`	at least **16**	**16**	**32**	**32**	**32**
`signed`						
`signed int`						
`unsigned`	`unsigned int`					
`unsigned int`						
`long`	`long int`	at least **32**	**32**	**32**	**32**	**64**
`long int`						
`signed long`						
`signed long int`						
`unsigned long`	`unsigned long int`					
`unsigned long int`						
`long long`	`long long int` (C++11)	at least **64**	**64**	**64**	**64**	**64**
`long long int`						
`signed long long`						
`signed long long int`						
`unsigned long long`	`unsigned long long int` (C++11)					
`unsigned long long int`						

Примитивные типы

Целочисленные знаковые

`float`	4B
`double`	8B
`long double`	10B





Логические

`bool`	1B

Безтиповые

`void`

Использование примитивных типов

Объявление переменной

```
type name = DefaultValue;
```

```
int x = 5;
```

```
bool enabled = true;
```

```
// указатель
int *c;

char *str = “qwerty\0”;
```

```
char a = 1,
     b = 2;

int c = a + b;
```

Использование примитивных типов

Объявление функции

```
ReturnType FunctionName(Type param1, Type param2…, Type paramN);
```

```
int sum(int a, int b);
```

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

int main() {
    printf("%d + %d = %d", 1, 2, sum(1, 2));
    printf("%d + %d = %d", 5, -12, sum(5, -12));
    printf("%d + %d = %d", 100, 0, sum(100, 0));

    return 0;
}
```

Использование примитивных типов

Арифметические операции

```
int a = 1, b = 2, c;

c = a + b;              // сложение                [c = 3]
c = a - b;              // вычитание               [c = -1]
c = a * b;              // умножение               [c = 2]
c = a / b;              // деление                 [c = 0]
c = a % b;              // остаток от деления      [c = 5]

c = a++;                // постинкремент    [a = 2, c = 1]
c = ++a;                // преинкремент     [a = 2, c = 2]

c = b--;                // постдекремент    [b = 1, c = 2]
c = --b;                // предекремент     [b = 1, c = 1]
```

Логические операции

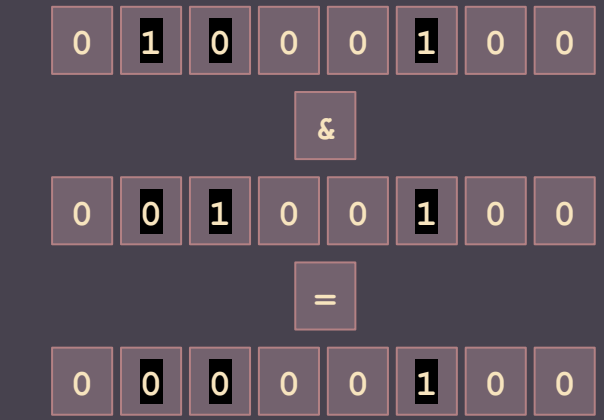
```
int a = 1;
bool b = false;
bool c = true;
bool res;

res = a && b;              // Логическое И
res = a || b;              // Логическое ИЛИ
res = !a;                  // Логическое НЕ

res = a == 1;              // Сравнение РАВНО
res = a != 1;              // Сравнение НЕРАВНО
res = a > 0;               // Сравнение БОЛЬШЕ
res = a < 2;               // Сравнение МЕНЬШЕ
res = a >= 1;              // Сравнение БОЛЬШЕ или РАВНО
res = a <= 1;              // Сравнение МЕНЬШЕ или РАВНО
```

Побитовые логические операции

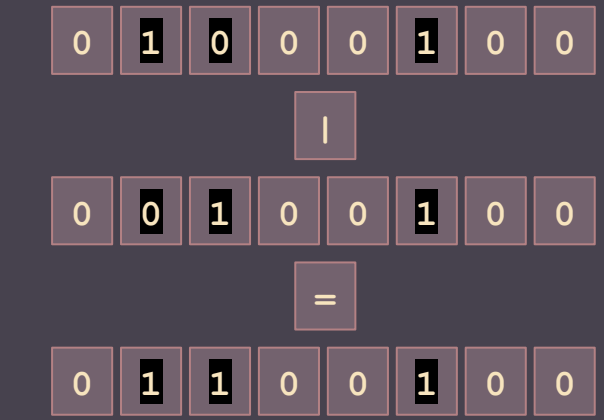
Побитовое И




```
c = a & b;        // Логическое И
```

Побитовые логические операции

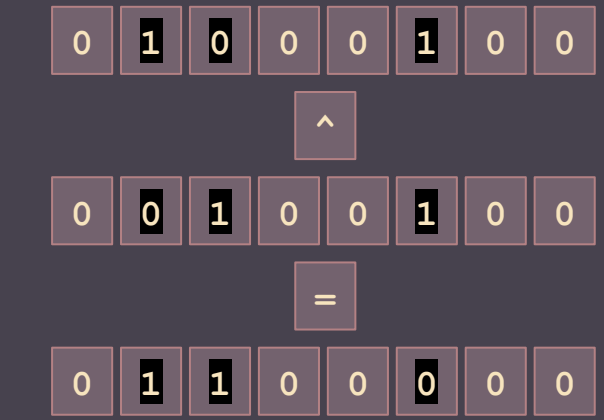
Побитовое ИЛИ




```
c = a | b;        // Логическое ИЛИ
```

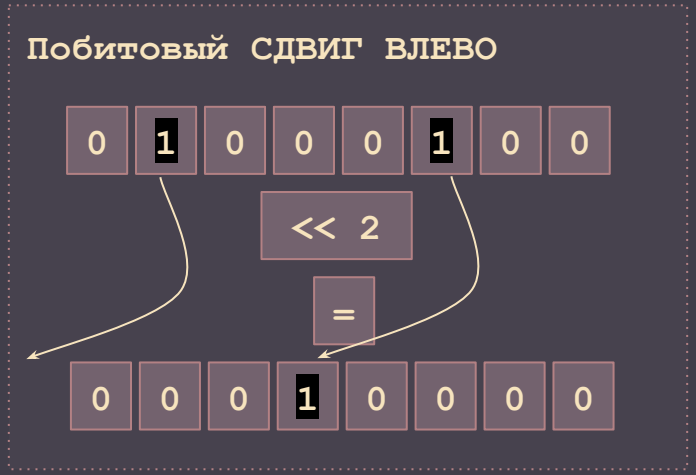
Побитовые логические операции

Побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ



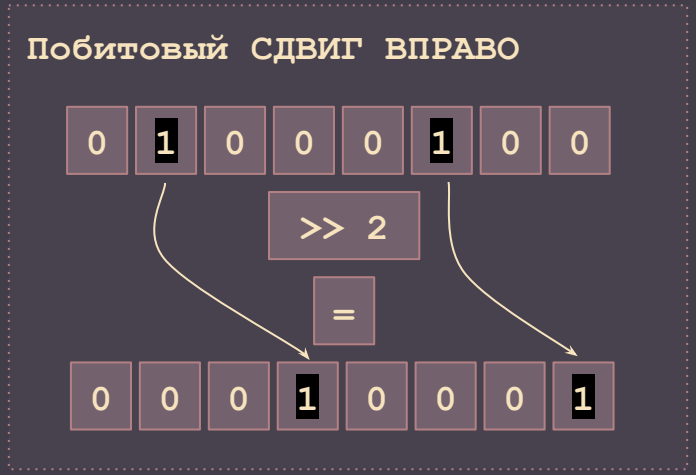

```
c = a ^ b;        // Логическое ИСКЛЮЧАЮЩЕЕ ИЛИ
```

Побитовые логические операции




```
c = a << 2;       // Логическое СДВИГ ВЛЕВО
```

Побитовые логические операции




```
c = a >> 2;       // Логическое СДВИГ ВПРАВО
```

Упрощенные операции

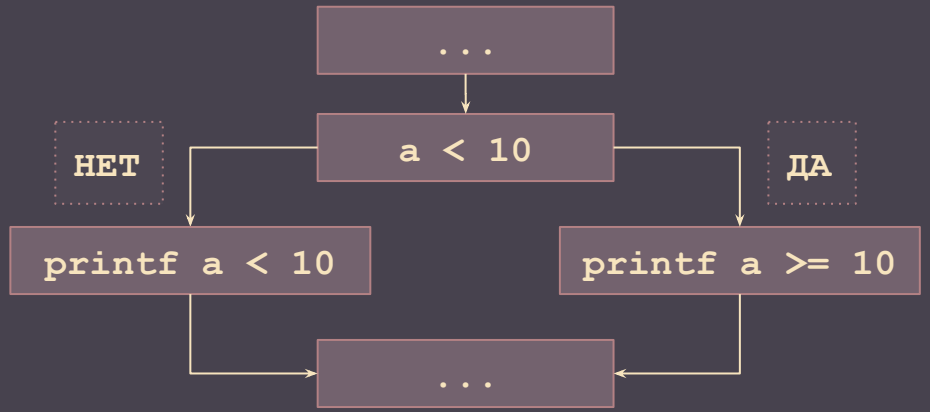
```
a += b;
a -= b;
a *= b;
a /= b;
a %= b;
a &= b;
a |= b;
a ^= b;
a <<= b;
a >>= b;
```

=

```
a = a + b;
a = a - b;
a = a * b;
a = a / b;
a = a % b;
a = a & b;
a = a | b;
a = a ^ b;
a = a << b;
a = a >> b;
```

Операция ветвления

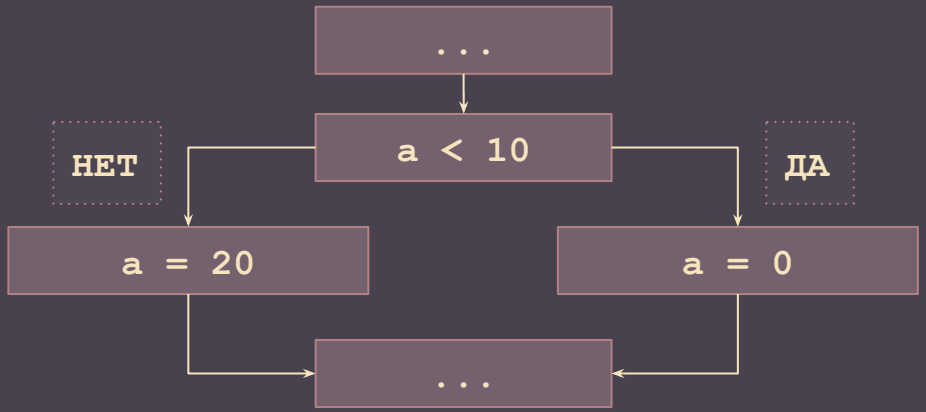
IF




```
...
if (a < 10) {
    printf(“a < 10”);
} else {
    printf(“a >= 10”);
}
...
```

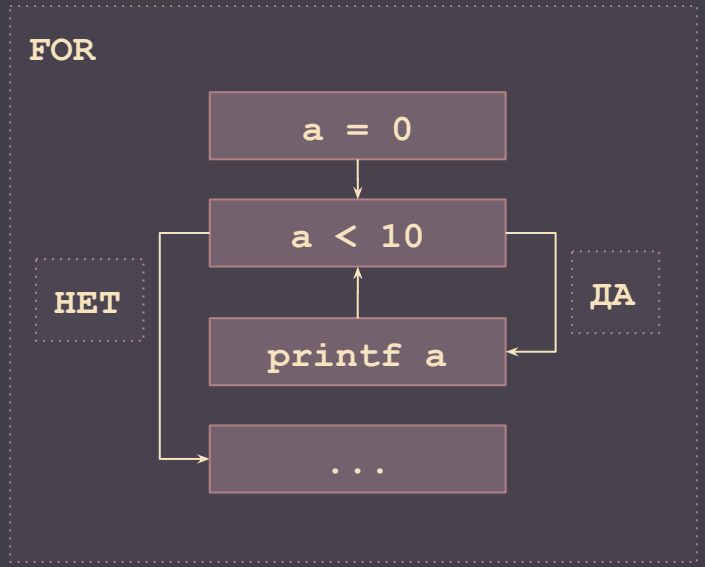
Операция ветвления

```
(логическое выражение) ? (выражение для случая ДА) : (выражение для случая НЕТ);
```




```
...
(a < 10) ? a = 0 : a = 20;
...
```

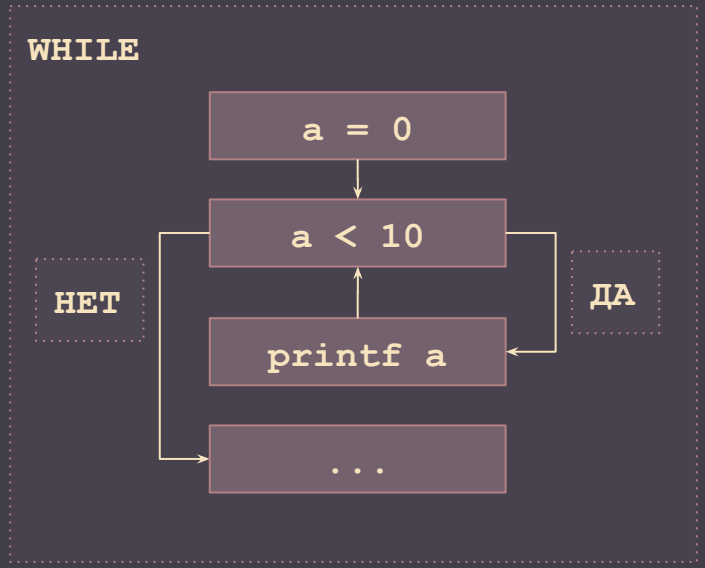
Циклические операции




```
for (int a = 0; a < 10; ++a) {
    printf("a = %d", a);
}
...
```

```
> a = 0
> a = 1
> a = 2
> a = 3
> a = 4
> a = 5
> a = 6
> a = 7
> a = 8
> a = 9
```

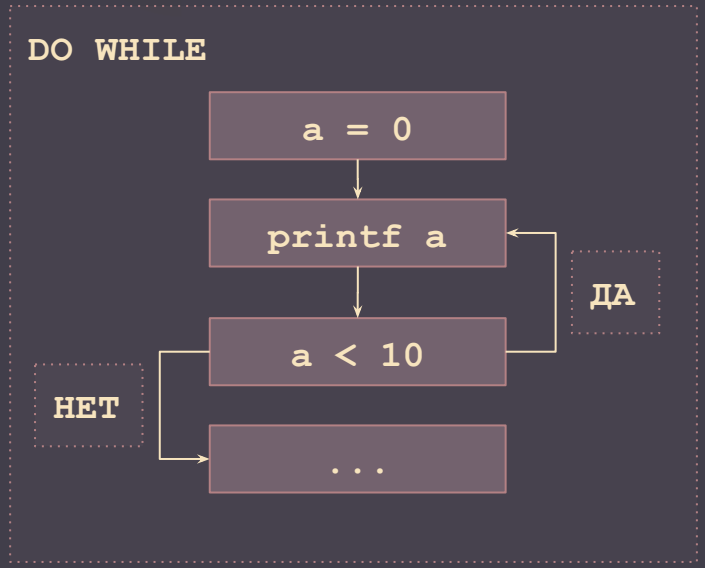
Циклические операции




```
int a = 0;
while (a < 10) {
    ++a;
    printf("a = %d", a);
}
...
```

```
> a = 0
> a = 1
> a = 2
> a = 3
> a = 4
> a = 5
> a = 6
> a = 7
> a = 8
> a = 9
```

Циклические операции




```
int a = 0;
do {
    ++a;
    printf("a = %d", a);
} while (a < 10);
...
```

```
> a = 0
> a = 1
> a = 2
> a = 3
> a = 4
> a = 5
> a = 6
> a = 7
> a = 8
> a = 9
> a = 10
```

Досрочный выход из цикла

```
int a = 0;
while (a < 10) {
    ++a;
    if ( (a % 2) == 0 ) {
        continue;
    }
    printf("a = %d", a);
}
...
```

```
> a = 1
> a = 3
> a = 5
> a = 7
> a = 9
```

Досрочный выход из цикла

```
int a = 0;
while (a < 10) {
    ++a;
    if ( a == 5 ) {
        break;
    }
    printf("a = %d", a);
}
...
```

```
> a = 0
> a = 1
> a = 2
> a = 3
> a = 4
```

Приоритеты операций

1	2	3	4	5	6	7	8	9	10	11	12	13
:: [] () . ->	a++ a--	++a --a	* / %	+ -	>> <<	< <= > >=	== !=	&&	||	?:	= *= /= %= += -=	,