Язык C++

Типы данных, идентификаторы, операторы, операторы ветвления, циклы, функции

Hello world

```
#include <iostream>

int main(int argc, char** argv) {
    std::cout << "Hello, world!\n";

    return 0;
}
```

Идентификаторы

- Конструкции и элементы программы нужно как-то называть
- Идентификаторы – это имена, используемые для обозначения переменных, типов, функций, шаблонов и т.д.
- Идентификаторы могут являться частью выражений (например c = a + b)

Идентификаторы

1. Буквы, цифры и “_”
2. Первый символ - буква или “_”
3. Прописные и строчные различаются
4. Не могут совпадать с ключевыми словами

Code Style

- Венгерская нотация
- camelCase
- snake_case
- PascalCase
- https://google.github.io/styleguide/cppguide.html

Встроенные типы данных

- char
- Целочисленные
  - int
  - short (int)
  - long (int)
- С плавающей точкой
  - float
  - double
- bool
- void
- nullptr_t

Модификаторы

- short
- long
- signed
- unsigned

Размеры и диапазоны для большинства 32-битных систем

Name	Size	Range
char	1byte	signed: -128 to 127 unsigned: 0 to 255
short	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)

numeric_limits

```
#include <iostream>
#include <limits>


int main(int argc, char** argv) {
   std:: cout << "Max value: " << std::numeric_limits<long>().max() << std::endl;
   std:: cout << "Min value: " << std::numeric_limits<double>().min() << std::endl;
   std:: cout << "Is signed value: " << std::numeric_limits<char>().is_signed << std::endl;

   return 0;
}
```

Целочисленные типы

#include <cstdint>

- `int8_t, int16_t,int32_t, int64_t`
- `uint8_t, uint16_t, uint32_t, uint64_t`

```
1 == sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long) <= sizeof(long long)
```

Целочисленные литералы

```
#include <iostream>

int main(int argc, char** argv) {
   int a = 162;
   int b = 0242; // OCT
   int c = 0xA2; // HEX
   int d = 0b010100010; // BIN

   std::cout << "a = " << a << std::endl
             << "b = " << b << std::endl
             << "c = " << c << std::endl
             << "d = " << d << std::endl;
   return 0;
}
```

Вещественные литералы

```
#include <iostream>

int main(int argc, char** argv) {
   double      a = 0.15;
   float       b = 0.15f;
   long double c = 15e-2l;
   float       d = 15e-2f;

   std::cout << "a = " << a << std::endl
             << "b = " << b << std::endl
             << "c = " << c << std::endl
             << "d = " << d << std::endl;
   return 0;
}
```

Представление чисел в памяти

- Целые числа
  - Прямой код
  - Обратный код
  - Дополнительный код
- Вещественные
  - Знак, порядок, мантисса

Символьные литералы

- Символьный литерал - **‘x’**
- Некоторые символьные литералы начинаются с эскейп-последовательности **‘\n’**
  - символ новой строки **‘\n’**
  - горизонтальная табуляция **‘\t’**
  - обратная слеш **‘\\’**
  - одиночная кавычка **’\“**
- Строковый литерал **“Hello \’ world\’\n”**

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Строковые литералы

- Строка - массив символов
- В конце спец символ конца строки **‘\0’**

bool

- true
- false

Enum (перечислимый тип)

```
enum Color  {
   RED,
   GREEN,
   BLUE
};

Color color = Color::BLUE;
```

Объявление переменных (declaration)

```
int a;
float b;
char c;

int k, l, n;

unsigned short s;
signed int i;
```

Определение переменных (definition)

```
int a     = 0;
double r = 1.23;
float b   = 23.5f;
float c   = 1.0e-3;
char ch   = 'c';
long l    = 23456789L;
int i     = 0X1F;
long double ld = 1.2345l;
```

Операторы

- Арифметические (+, -, *, /, %)
- Сравнение (>, >=, <, <=, ==, !=)
- Логические (&&, ||)
- Инкремента и Декремента (++, --)
- Побитовые (&, |, ^, <<, >>, ~)
- Присваивание (=, +=, *=,)
- Условный (?:)
- Специальные (sizeof, static_cast, ...)

Преобразования типов

- **Неявные преобразования**
  - Если какой-либо из операндов принадлежит типу **long double**, то и другой приводится к **long double**.
  - В противном случае, если какой-либо из операндов принадлежит типу **double**, то и другой приводится к **double**.
  - В противном случае, если какой-либо из операндов принадлежит типу **float**, то и другой приводится к **float**.
  - В противном случае операнды типов **char** и **short** приводятся к **int**.
  - И наконец, если один из операндов типа **long**, то и другой приводится к **long**.
- **Явное преобразование (c-style cast)** *(тип) переменная*

sizeof

```
int x;
printf("sizeof(int) = %zu\n", sizeof(int));
printf("sizeof(float) = %zu\n", sizeof(float));
printf("sizeof(char) = %zu\n", sizeof(char));
printf("sizeof(long long) = %zu\n", sizeof(long long));

printf("sizeof(x) = %zu\n", sizeof(x));
```

a[k]	индексы	16	слева направо
f(...)	вызов функции	16	слева направо
.	прямой выбор	16	слева направо
->	опосредованный выбор	16	слева направо
++ --	положительное и отрицательное приращение	16	слева направо
++ --	положительное и отрицательное приращение	15	справа налево
sizeof	размер	15	справа налево
~	побитовое НЕ	15	справа налево
!	логическое НЕ	15	справа налево
- +	изменение знака, плюс	15	справа налево
&	адрес	15	справа налево

*	опосредование (разыменование)	15	справа налево
(имя типа)	приведение типа	14	справа налево
* / %	мультипликативные операции	13	слева направо
+ -	аддитивные операции	12	слева направо
<< >>	сдвиг влево и вправо	11	слева направо
< > <= >=	отношения	10	слева направо
== !=	равенство/неравенство	9	слева направо
&	побитовое И	8	слева направо
^	побитовое исключающее ИЛИ	7	слева направо
|	побитовое ИЛИ	6	слева направо
&&	логическое И	5	слева направо
||	логическое ИЛИ	4	слева направо
? :	условие	3	справа налево
= += -= *= /= %= <<= >>= &= ^= |=	присваивание	2	справа налево
,	последовательная оценка	1	слева направо

```
a + b * c << d || 25 != 32 && !c++
```

Оператор

- Оператор заканчивается ***«;»***

  c = a + b;

  printf(“Hello World”);

  i++;

- ***{оператор1; оператор2; оператор3;}*** **-** составной оператор

if-else

```
if (выражение)
    оператор1 ( или составной оператор);
else
   оператор2 ( или составной оператор);
```

- else-часть может отсутствовать
- if(выражение != 0) тоже самое что if(выражение)

if-else

```
// n == -2, a = 1, b = 2, z = 20;
if (n > 0)
     if(a > b)
          z = a;
 else
     z = b;
```

```
// n == -2, a = 1, b = 2, z = 20;
if (n > 0)
     if(a > b)
          z = a;
     else
          z = b;
```

if-else

```
// n == -2, a = 1, b = 2, z = 20;
if (n > 0) {
     if(a > b) {
          z = a;
     }
}
else {
     z = b;
}
```

```
// n == -2, a = 1, b = 2, z = 20;
if (n > 0) {
     if(a > b) {
          z = a;
     }
     else {
          z = b;
     }
}
```

else-if

```
if (выражение1)
    оператор1;
else if(выражение2)
   оператор2;
else if(выражение3)
   оператор3;
else if(выражение4)
   оператор4;
else
   оператор5;
```

Цикл while

while (выражение)
оператор

```
int a;
   std::cin >> a;

   while(a > 0) {
       std::cout << a << std::endl;
      --a;
   }
```

Цикл do-while

do
оператор;
while (выражение);

```
unsigned long n;
do
{
    std::cout << "Enter number (0 to end): ";
    std::cin >> n;
    std::cout << "You entered: " << n << std::endl;
} while (n != 0);
```

Цикл for

```
for (выр1; выр2; выр3)
    оператор
```

```
выр1;
while (выр2)
{
    оператор
    выр3;
}
```

Цикл for

for (инициализация; условие; инкремент)
оператор;

```
for (int n = 10; n > 0; n--) {
     std::cout << n << ", ";
}


int i;
for (int n = 0, i = 100 ; n != i; n++, i--) {
  std::cout << n << ", ";
}
```

Цикл range for

for (объявление : диапазон) оператор;

```
for (int n : {0, 1, 2, 3, 4, 5}) {
     std::cout << n << ' ';
}
std::cout << '\n';
```

Операторы break, continue

```
for (int n = 10; n > 0; n--){
    std::cout << n << ", ";
    if (n == 3){
        std::cout << "countdown aborted!";
        break;
    }
}
```

Операторы break, continue

```
for (int n = 10; n > 0; n--) {
  if (n == 5)
      continue;
  std::cout << n << ", ";
}
```

Оператор switch

```
switch (выражение)
{
    case константа1:
     группа операторов1;
     break;
   case constant2:
     группа операторов2;
     break;
   . . .
   default:
     группа операторов по умолчанию;
}
```

Оператор switch

- Константы – целые
- Вычисления начинаются с первой совпавшей с константой в ветке и выражения
- Все константы должны быть разные
- Если совпадения не нашлось то выполняется ***default***
- ***Break*** вызывает выход из switch
- Сквозное выполнение

Оператор switch

```
switch (x){
  case 1:
      printf("x is 1");
      break;
  case 2:
      printf("x is 2");
      break;
  default:
      printf("unknown");
}
```

```
if (x == 1) {
  printf("x is 1");
}
else if (x == 2){
  printf("x is 2");
}
else {
  printf("unknown");
}
```

Оператор switch

```
switch (x){
  case 1:
  case 2:
  case 3:
      printf("x is 1, 2 or 3");
      break;
  default:
      printf ("x is not 1, 2, 3");
}
```

Функции

```
тип имя(параметр1, параметр2) {
    объявления и инструкции
}
```

Функции

```
int addition (int a, int b) {
  int result;
  result = a + b;
  return result;
}

int main () {
  int z;
  z = addition (5, 3);
  return 0;
}
```

Функции. **declarations & definitions**

```
int max(int a, int b); // declaration

int main() {
  int c = max(10, 2);  // ok
  max(1);              // compile-time error
  return 0;
}

int max(int a, int b) { // definition
  return a > b ? a : b;
}
```

Функция без возвращаемого значения

```
#include <iostream>

void printmessage() {
  std::cout << "I'm a function!\n";
}

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

Нет возвращаемого значения

Объявление и определение в одном месте

main

- int **main (void) {** *…* **}**
- int **main (**int *argc*, char **argv*[]**) {** *...* **}**
- int **main (**int *argc*, char **argv*[] , *other_parameters* **)**
  { *...* **}**

  EXIT_SUCESS, EXIT_FAILURE

main

```
#include <iostream>

int main(int argc, char* argv[]) {
  for(int i = 0; i < argc; ++i)
      std::cout << argv[i] << " ";

  return 0;
}
```

argc - размер массива
argv - массив строк

Рекурсия

```
#include <iostream>

unsigned long long factorial(unsigned int n) {
  if (n == 0)
      return 1;
  else return n * factorial(n-1);
}

int main() {
  std::cout << factorial(5) << std::endl;

  return  0;
}
```

Рекурсия

```
#include <iostream>

unsigned long long factorial(unsigned int n) {
  unsigned long long result = 1;
  for (int i = 2; i <= n;++i) {
      result *= i;
  }
  return  result;
}

int main() {
  std::cout << factorial(5) << std::endl;

  return  0;
}
```

«Затемнение» внешних переменных

```
#include <iostream>

int x;
int y;
void func(double x) {
  double y;
  std::cout << "x = " << x << " y = " <<  y << std::endl;
}
int main() {
  x = 21;
  {
      int x = 10;
      y = 239;
      std::cout << "x = " << x << " y = " <<  y << std::endl;
      func(y);
  }
  std::cout << "x = " << x << " y = " <<  y << std::endl;
  return 0;
}
```