Модуль 3

- Простые типы данных
- Размер типов
- Циклы
- Арифметические операторы
- Приоритет операций

Простые типы данных

Как хранятся данные в компьютере

1. Двоичная система счисления
2. 1 байт = 8 бит
3. Машинное слово зависит от архитектуры процессора

Объявление:

```
char c;
short s;
int i = 5;
long l = 7;
long long ll;
doble d;
```

Целочисленные типы данных:

	Тип	**Минимальный размер**
Символьный тип данных	char	1 байт
Целочисленный тип данных	short	2 байта
	int	2 байта (но чаще всего 4 байта)
	long	4 байта
	long long	8 байт

CHAR

Переменная типа char фактически хранит числовое значение символа, а не сам символ.

ASCII (сокр. от *«**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange»*) — это американский стандартный код для обмена информацией, который определяет способ представления символов английского языка

```
char ch(5); // инициализация переменной типа char целым числом 5
char ch('5'); // инициализация переменной типа char символом '5' (53)
```

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

CHAR

Существуют следующие кодировки Unicode:

UTF-32 — требует 32 бита для представления символа.

UTF-16 — требует 16 бит для представления символа.

UTF-8 — требует 8 бит для представления символа.

CHAR. Управляющие символы

Название	**Символ**	**Значение**
Предупреждение (**a**lert)	\a	Предупреждение (звуковой сигнал)
Backspace	\b	Перемещение курсора на одну позицию назад
formfeed	\f	Перемещение курсора к следующей логической странице
Символ новой строки (**n**ewline)	\n	Перемещение курсора на следующую строку
Возврат каретки (carriage **r**eturn)	\r	Перемещение курсора в начало строки
Горизонтальный таб (horizontal **t**ab)	\t	Вставка горизонтального TAB
Вертикальный таб (**v**ertical tab)	\v	Вставка вертикального TAB
Одинарная кавычка	\’	Вставка одинарной кавычки (или апострофа)
Двойная кавычка	\”	Вставка двойной кавычки
Бэкслеш	\\	Вставка обратной косой черты (бэкслеша)

CHAR.

Что использовать: ‘\n’ или std::endl?

Диапазоны

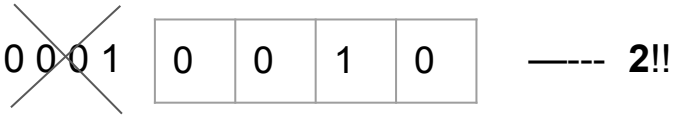
Размер/Тип	**Диапазон значений**
1 байт signed	от -128 до 127
1 байт unsigned	от 0 до 255
2 байта signed	от -32 768 до 32 767
2 байта unsigned	от 0 до 65 535
4 байта signed	от -2 147 483 648 до 2 147 483 647
4 байта unsigned	от 0 до 4 294 967 295
8 байтов signed	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
8 байтов unsigned	от 0 до 18 446 744 073 709 551 615

signed, unsigned, переполнение.

По умолчанию все типы signed. Следует использовать беззнаковые переменные только в том случае если это необходимо.

Переполнение

18 —- 0001 0010



—---- **2**!!

Целочисленные типы фиксированного размера

Название	**Тип**	**Диапазон значений**
int8_t	1 байт signed	от -128 до 127
uint8_t	1 байт unsigned	от 0 до 255
int16_t	2 байта signed	от -32 768 до 32 767
uint16_t	2 байта unsigned	от 0 до 65 535
int32_t	4 байта signed	от -2 147 483 648 до 2 147 483 647
uint32_t	4 байта unsigned	от 0 до 4 294 967 295
int64_t	8 байт signed	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
uint64_t	8 байт unsigned	от 0 до 18 446 744 073 709 551 615

Типы данных с плавающей точкой

Тип	Минимальный размер	Типичный размер
float	4 байта	4 байта
double	8 байт	8 байт
long double	8 байт	8, 12 или 16 байт

- `int n{5};` *// 5 – это целочисленный тип*
- `double d{5.0};` *// 5.0 – это тип данных с плавающей точкой (по умолчанию double)*
- `float f{5.0f};` *// 5.0 – это тип данных с плавающей точкой ("f" от "float")*

Хранение в памяти чисел с плавающей запятой

- **Экспоненциальная запись** очень полезна для написания длинных чисел в краткой форме. Числа в экспоненциальной записи имеют следующий вид: **`мантисса × 10`$^{\textbf{\textit{экспонент}}}$**. Например, рассмотрим **выражение** `1.2 × 10`^{4}. Значение `1.2` — это **мантисса** (или ***«значащая часть числа»***), а `4` — это **экспонент** (или ***«порядок числа»***). Результатом этого выражения является значение 12000.
- Масса электрона равна `9.1093822e-31 кг`

Сравнение чисел с плавающей запятой

- **fabs(a - b) < std::numeric_limits<double>::epsilon()**
- Использование сторонних библиотек, таких как **<boost/test/floating_point_comparison.hpp>**

Логический тип данных bool

Логические переменные — это переменные, диапазон которых состоит только из двух возможных значений: `true` (`1`) и `false` (`0`).

```
1.  bool b1 = true; // копирующая инициализация
2.  bool b2(false); // прямая инициализация
3.  bool b3 { true }; // uniform-инициализация (C++11)
4.  b3 = false; // операция присваивания
5.  bool b1 = !true; // значение b1 - false
6.  bool b2(!false); // значение b2 - true
```

Циклы

Цикл while

```
while (условие)
    тело цикла;
```

`while(true)`
`{}` - **бесконечный цикл**
Единственный способ выйти из бесконечного цикла — использовать операторы **return**, **break**, **goto**, выбросить исключение или воспользоваться **функцией exit()**.

Переменных лучше определять перед циклом

Циклы

Цикл do while

```
do
    тело цикла;
while (условие);
```

Циклы

Цикл for

```
for (объявление переменных; условие; инкремент/декремент счетчика)
    тело цикла;
```

Цикл for в C++ выполняется в 3 шага:

***Шаг №1**:* **Объявление переменных**. Как правило, здесь выполняется определение и инициализация счетчиков цикла, а точнее — одного счетчика цикла. Эта часть выполняется только один раз, когда цикл выполняется впервые.

***Шаг №2**:* **Условие**. Если оно равно false, то цикл немедленно завершает свое выполнение. Если же условие равно true, то выполняется тело цикла.

***Шаг №3**:* **Инкремент/декремент счетчика цикла**. Переменная увеличивается или уменьшается на единицу. После этого цикл возвращается к шагу №2.

Циклы

В цикле for необходимо уделять внимание условию, поскольку условие может содержать > или >=, а счетчик инициализируется 0.

```
for (;;)
    тело цикла; - бесконечный цикл
```

Циклы

Цикл foreach

```
for (объявление_элемента : массив)
   Стейтмент;
```

Циклы foreach не предоставляют прямой способ получения индекса текущего элемента массива.

Циклы

break, return, continue

1. Оператор break завершает работу цикла, а выполнение кода продолжается с первого стейтмента, который находится сразу же после этого или цикла
2. Оператор return завершает выполнение всей функции, в которой находится цикл, а выполнение продолжается в точке после вызова функции
3. Оператор continue позволяет сразу перейти в конец тела цикла, пропуская весь код, который находится под ним.

Многие учебники рекомендуют не использовать операторы break и continue!!!

Арифметические операторы

Унарные арифметические операторы — это операторы, которые применяются только к одному операнду. Существуют два унарных арифметических оператора: плюс (`+`) и минус (`-`).

Оператор	**Символ**	**Пример**	**Операция**
Унарный плюс	+	+x	Значение x
Унарный минус	−	−x	Отрицательное значение x

Арифметические операторы

Бинарные арифметические операторы — это операторы, которые применяются к двум операндам (слева и справа). Существует 5 бинарных операторов.

Оператор	**Символ**	**Пример**	**Операция**
Сложение	+	x + y	x плюс y
Вычитание	−	x − y	x минус y
Умножение	*	x * y	x умножить на y
Деление	/	x / y	x разделить на y
Деление с остатком	%	x % y	Остаток от деления x на y

Арифметические операторы

Арифметические операторы присваивания

Оператор	Символ	Пример	Операция
Присваивание	=	x = y	Присваиваем значение y переменной x
Сложение с присваиванием	+=	x += y	Добавляем y к x
Вычитание с присваиванием	−=	x −= y	Вычитаем y из x
Умножение с присваиванием	*=	x *= y	Умножаем x на y
Деление с присваиванием	/=	x /= y	Делим x на y
Деление с остатком и с присваиванием	%=	x %= y	Присваиваем остаток от деления x на y переменной x

Арифметические операторы

Оператор возведения в степень

`pow(base, exponent)` эквивалентно $base^{exponent}$

Приоритет операций

https://en.cppreference.com/w/cpp/language/operator_precedence

"Associativity Left-to-right" означает, что операции с одинаковым приоритетом будут выполняться слева
направо. Другими словами, если у вас есть несколько операций с одинаковым приоритетом в выражении, то они будут вычисляться в порядке, указанном слева направо.