Базовые конструкции языка C++

Пример для разминки

```
#include <iostream>

int main()
{
    using namespace std;

    double dollars;
    const double DOLLARS_TO_RUBLES = 58.1868963;

    cout << "Enter amount of money ($): ";
    cin >> dollars;

    double rubles = dollars * DOLLARS_TO_RUBLES;

    cout << "You have " << dollars << "$" << endl << ":)" << endl;
    cout << "And " << rubles << " rubles\n";
    cout << "Goodbye!\n";
}
```

Алфавит языка C++

- Базовый набор допустимых символов состоит из 96 символов:
  - Символ пробела
  - Горизонтальная / вертикальная табуляция
  - “form feed”
  - Символ перевода каретки

А также 91 символ с графическим представлением:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
_ { } [ ] # ( ) < > % : ; . ? * + - / ^ & | ~ ! = , \ " '
```

Типы данных

Логический тип: bool – тип, способный хранить одно из двух значений (true или false).

Символьные типы:

- signed char— тип для знакового представления символов.
- unsigned char — тип для беззнакового представления символов.
- char - тип для представления символов, который может наиболее эффективно обрабатываться в целевой системе.
- wchar_t — тип для широкого представления символов.
- char16_t — тип для представления символов в UTF-16. (Начиная с C++11)
- char32_t — тип для представления символов в UTF-32. (Начиная с C++11)

Типы данных

Целочисленный тип: int — базовый целочисленный тип. Может быть опущен, если представлен любой из модификаторов. Если не представлен ни один из модификаторов размера, гарантировано имеет ширину не меньше 16 бит. Тем не менее, на 32/64-битных системах почти всегда имеет ширину не меньше 32 бит.

Тип void: void – не полный тип, у которого не существует ни одного значения.

Модификаторы типов

Модификаторы используются для изменения целочисленного типа. Могут быть использованы в любом порядке. Только один модификатор каждой группы может быть представлен в имени типа.

Знаковость

signed – целевой тип будет иметь знаковое представление (исп. по умолчанию)

unsigned – целевой тип будет иметь беззнаковое представление.

Размер

short - целевой тип будет оптимизирован по размеру и иметь ширину не меньше 16 бит.

long - целевой тип будет иметь ширину не меньше 32 бит.

long long - целевой тип будет иметь ширину не меньше 64 бит.

Типы с плавающей точкой

- float — тип с плавающей точкой одинарной точности. Обычно 32-битный тип с плавающей точкой формата IEEE-754
- double — тип с плавающей точкой двойной точности. Обычно 64-битный тип с плавающей точкой формата IEEE-754
- long double — тип с плавающей точкой повышенной точности. Не обязательно отображается на типы IEEE-754.

Сводная таблица типов

Спецификатор типа	Эквивалентный тип	Ширина в битах согласно модели данных				
		Стандарт C++	LP32	ILP32	LLP64	LP64
`short`	`short int`	не меньше чем **16**	**16**	**16**	**16**	**16**
`short int`						
`signed short`						
`signed short int`						
`unsigned short`	`unsigned short int`					
`unsigned short int`						
`int`	`int`	не меньше чем **16**	**16**	**32**	**32**	**32**
`signed`						
`signed int`						
`unsigned`	`unsigned int`					
`unsigned int`						
`long`	`long int`	не меньше чем **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)	не меньше чем **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`						

Правила назначения имен идентификаторов

- Не может содержать пробелов
- Может начинаться с символов a-z, A-Z, _
- Символы кроме первого могут дополнительно содержать 0-9
- Не может быть ключевым словом языка C++
- Имена чувствительны к регистрам
- Идентификаторам необходимо назначать только осмысленные имена
- Нельзя именовать идентификатор с двух нижних подчеркиваний и одного подчеркивания и последующей заглавной буквой

Ключевые слова C++

alignas	const_cast	for	public	thread_local
alignof	continue	friend	register	throw
asm	decltype	goto	reinterpret_cast	true
auto	default	if	requires	try
bool	delete	inline	return	typedef
break	do	int	short	typeid
case	double	long	signed	typename
catch	dynamic_cast	mutable	sizeof	union
char	else	namespace	static	unsigned
char16_t	enum	new	static_assert	using
char32_t	explicit	noexcept	static_cast	virtual
class	export	nullptr	struct	void
concept	extern	operator	switch	volatile
const	false	private	template	wchar_t
constexpr	float	protected	this	while

Примеры объявления

```
int var;
double 4ever;
unsigned long long int ____;
char my symbol;
float while;
bool isCharacter = false;
int Name;
int name;
```

Адрес переменной

Для взятия адреса переменной используется оператор амперсанд &

```
#include <iostream>

int main()
{
    int variable = 1024;

    std::cout << "variable value: " << variable << std::endl;
    std::cout << "variable address: " << &variable << std::endl;
}
```

Оператор sizeof()

```
#include <iostream>

int main()
{
    int variable = 1024;

    std::cout << "variable takes "
              << sizeof(variable) << " bytes of memory"
              << std::endl;

    std::cout << "Type ‘int’ takes the same: " <<
                  sizeof(int) << std::endl;
}
```

Три вариации синтаксиса инициализации переменных

- `int variable = 12;`
- `int variable(12);`
- `int variable = { 12 };`
- `int variable { 12 };`

Операции над фундаментальными типами в языке C++

- Сложение (x + y)
- Вычитание (x – y)
- Умножение (x * y)
- Деление (x / y)
- Инкремент (++)
- Декремент (--)
- Остаток от деления (%) (для целочисленных)
- Логические операции (для bool)

Приоритеты операций C++

- 1. Постинкремент, постдекремент (++, --)
- 2. Прединкремент, преддекремент (++, --)
- 2. Унарный плюс и минус (+, -)
- 2. Логическое НЕ (!)
- 2. Приведение к типу (type)
- 2. Взятие адреса (&), sizeof()
- 3. Умножение, деление, взятие остатка
- 4. Сложение, вычитание
- 5. Операции сравнения (<, <=, >, >=)
- 6. Операции сравнения (==, !=)
- 7. Логическое И
- 8. Логическое ИЛИ
- 9. Оператор присваивания

Пример

```
#include <iostream>

int main()
{
    int leftNumber, rightNumber;

    std::cout << "Enter left number: ";
    std::cin >> leftNumber;
    std::cout << "Enter right number: ";
    std::cin >> rightNumber;

    int sum = leftNumber + rightNumber;
    int sub (leftNumber - rightNumber);
    int mul = { leftNumber * rightNumber };
    int div { leftNumber / rightNumber };
    int mod = leftNumber % rightNumber;

    std::cout << "results: " << std::endl;
    std::cout << "sum = " << sum << std::endl;
    std::cout << "sub = " << sub << std::endl;
    std::cout << "mul = " << mul << std::endl;
    std::cout << "div = " << div << std::endl;
    std::cout << "mod = " << mod << std::endl;
}
```

Тип bool и логические операции

Над типом bool дополнительно введены следующие операции:

- Логическое И (&&)
- Логическое ИЛИ (||)
- Логическое НЕ (!)
- Сравнения (==), (!=), (>=), (<=), (<), (>)

```
bool boolean = 132 && (122 == 1);
bool boolean_2 = true || false;
bool boolean_3 = 1024 > 12 && 222 <= 222;
```

Литералы

- **Бинарные литералы:** 0b[двоичное число] или 0B[двоичное число]
- **Восьмеричные литералы:** 0[восьмеричное число]
- **Шестнадцатеричные литералы:** 0x[шестнадцатеричное число] или 0X[шестнадцатеричное число]

```
int v1 = 0xCAFE;
int v2 = 0b101010;
int v2 = 07712;
```

Литералы, уточняющие тип

- Суффикс unsigned типа: u U
- Суффикс long типа: l L
- Суффикс long long типа: ll LL
- Суффикс float типа: f F

```
std::cout << 1081u << 2000. << 1200.f;
std::cout << 12.31 << 12LL << 12l;
```

Управляющие последовательности

Название символа	Символ ASCII	Код C++	Десятичный код ASCII	Шестнадцате-ричный код ASCII
Новая строка	NL (LF)	\n	10	0xA
Горизонтальная табуляция	HT	\t	9	0x9
Вертикальная табуляция	VT	\v	11	0xB
Забой	BS	\b	8	0x8
Возврат каретки	CR	\r	13	0xD
Предупреждение	BEL	\a	7	0x7
Обратная косая черта	\	\\	92	0x5C
Знак вопроса	?	\?	63	0x3F
Одинарная кавычка	'	\'	39	0x27
Двойная кавычка	"	\"	34	0x22

Пример явного преобразования типов

```
int value1 = 1000;
int value2 = 333;

std::cout << value1 / value2 << std::endl;
std::cout << (double)value1 / value2 << std::endl;
std::cout << double(value1) / value2 << std::endl;
std::cout << static_cast <double> (value1) / value2 << std::endl;
```

Преобразование типов

- 1. Если один из операндов имеет тип long double, то другой операнд преобразуется в long double.
- 2. Иначе, если один из операндов имеет тип double, то другой операнд преобразуется в double.
- 3. Иначе, если один из операндов имеет тип float, то другой операнд преобразуется в float.
- 4. Иначе, операнды имеют целочисленный тип, поэтому выполняется целочисленное расширение.
- 5. В этом случае, если оба операнда имеют знак или оба операнда беззнаковые, и один из них имеет меньший ранг, чем другой, он преобразуется в больший ранг.
- 6. Иначе, один операнд имеет знак, а другой беззнаковый. Если беззнаковый операнд имеет больший ранг, чем операнд со знаком, последний преобразует в тип беззнакового операнда.
- 7. Иначе, если тип со знаком может представить все значения беззнакового типа, беззнаковый операнд преобразуется к типу операнда со знаком.
- 8. Иначе, оба операнда преобразуются в беззнаковую версию типа со знаком.