



C++

Ввод и вывод.
Условия.

Кузнецов Арсений Алексеевич

kuznetsov.misis@gmail.com

Библиотека для работы с ПОТОКОМ ВВОДА/ВЫВОДА

```
#include <iostream>
```

Вывод на экран

- Функция `cout`
- Оператор `<<`
- Оператор перевода строки `endl`
- Можно передать переменную, текст в двойных кавычках, перевод строки

```
cout << "Text" << myVar << "Enother text" << endl;
```

Получение данных

- Функция `cin`
- Оператор `>>`
- Записывает данные в переменную
- Не должно быть пробелов

```
cin >> myVar;
```

Получение строковых данных

- Функция `getline`
- Записывает данные в строковую переменную
- Можно передать пробелы

```
getline (cin, myVarString);
```

Контроль ошибок ввода данных

- `cin.good()` – проверяет, верны ли данные
- `cin.clear()` – очищает ошибки
- `cin.ignore()` – очищает оставшиеся данные в потоке



Условие

- Переменная, функция или операция над переменными, результатом которой является ИСТИНА или ЛОЖЬ

Логические операторы

&&	логическое И (AND)	<code>expr && expr</code>
	логическое ИЛИ (OR)	<code>expr expr</code>
==	равно	<code>expr == expr</code>
!=	не равно	<code>expr != expr</code>
>	больше	<code>expr > expr</code>
>=	больше или равно	<code>expr >= expr</code>
<	меньше	<code>expr < expr</code>
<=	меньше или равно	<code>expr <= expr</code>

Арифметические операторы

*	умножение	<code>expr * expr</code>
/	деление	<code>expr / expr</code>
%	остаток от деления (деление по модулю)	<code>expr % expr</code>
+	сложение (плюс)	<code>expr + expr</code>
-	вычитание (минус)	<code>expr - expr</code>

Присваивающие арифметические операторы

<code>*=</code>	умножение и присваивание	<code>lvalue *= expr</code>
<code>/=</code>	деление и присваивание	<code>lvalue /= expr</code>
<code>%=</code>	остаток и присваивание	<code>lvalue %= expr</code>
<code>+=</code>	сложенней присваивание	<code>lvalue += expr</code>
<code>-=</code>	вычитание и присваивание	<code>lvalue -= expr</code>

`a += b` аналогично `a = a + b`

Инкремент и декремент

- инкремент

`++lvalue` (возвращает новое значение)

`lvalue++` (возвращает старое значение)

- декремент

`--lvalue` (возвращает новое значение)

`lvalue--` (возвращает старое значение)

- `a++` аналогично `a = a + 1`

Условный оператор if

```
if ( УСЛОВИЕ )  
{  
    // если условие выполнено, работает этот блок кода  
}  
else  
{  
    // если условие не выполнено, работает этот блок кода  
}
```

Условный оператор switch

```
switch ( ПЕРЕМЕННАЯ )
{
    case 1: // если ПЕРЕМЕННАЯ равна 1
            // Выполнится этот блок кода
            break;
    case 2: // если ПЕРЕМЕННАЯ равна 2
            // Выполнится этот блок кода
            break;
    default: // Во всех остальных случаях
            // Выполнится этот блок кода
            break;
}
```

Пример использования `if`

```
if( cin.good() )
{
    cout << "Good! Correct value: " << someNum << "!";
}
else
{
    cout << "Error! Invalid value!";
}
```

Пример использования switch

```
switch (someNum % 3)
{
    case 0:
        cout << someNum << " is divided by 3." << endl;
        break;
    case 1:
    case 2:
        cout << someNum << " isn't divided by 3." << endl;
        break;
    default:
        assert(0);
        break;
}
```