

# Оператор за условен преход

В някои случаи искаме да изпълним даден код, само когато дадено условие е изпълнено. Това е възможно с операторите `if/else` или `switch`.

## Условен оператор `if`

Ако искаме даден блок от код (блок от код се дефинира чрез `{ }`) да се изпълни когато дадено условие е вярно, използваме оператора `if`:

Синтаксис:

```
if(условие)
{
    //Код който ще се изпълни
}
```

*Пример:*

Нека направим програма, в която потребителя въвежда число, а програмата казва дали числото е положително (приемаме че и 0-та е положително число)

```
#include <iostream>
using namespace std;

int main()
{
    int user_number = 0;

    cout << "Please enter a number";
    cin >> user_number;

    if(user_number >= 0)
    {
        cout << "The number is positive" << endl;
    }

    return 0;
}
```

```
cpp_playground : bash — Konsole
Please enter a number
15
The number is positive
01:44:29 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

```
cpp_playground : bash — Konsole
Please enter a number
-3
01:44:45 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Ако искаме повече от едно условие да бъде изпълнени **едновременно**, може да използваме оператор `&&` (логическо **и**) между всеки израз:

```
if(условие1 && условие2 && условие3 && ... && условиеN)
```

Ако искаме **поне едно** от няколко условия да бъдат изпълнени, може да използваме `||` (логическо **или**) между всеки израз:

```
if(условие1 || условие2 || условие3 || ... || условиеN)
```

*Пример*

```
#include <iostream>
using namespace std;

int main()
{
    int dice = 1;

    cout << "Throw a dice and enter the result ";

    cin >> dice;
```

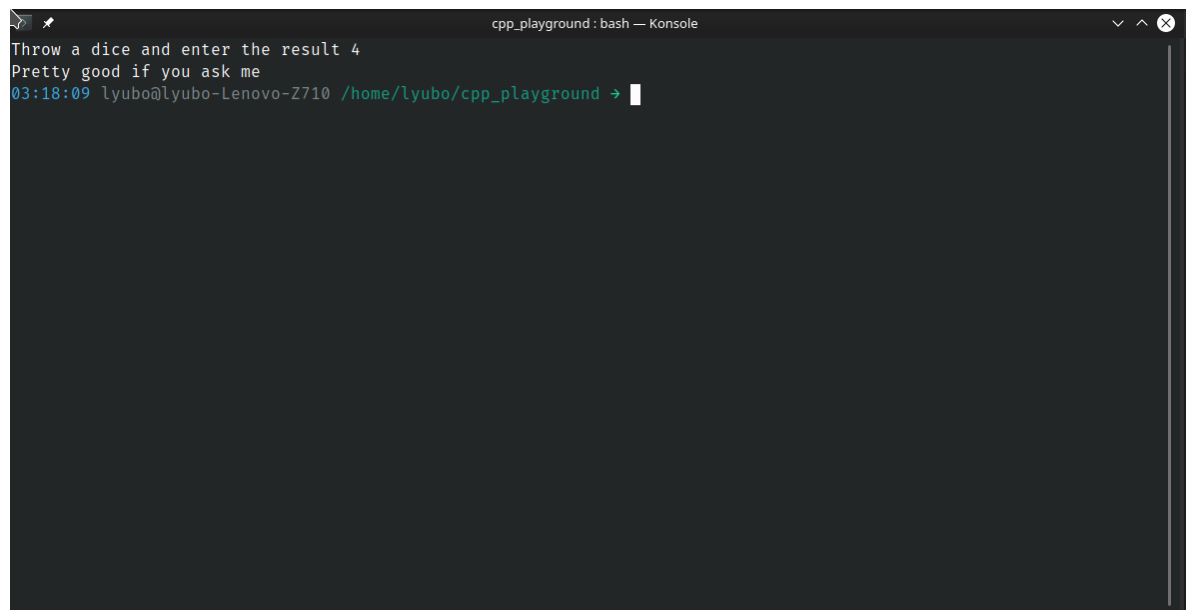
```
if(dice < 1 || dice > 6)
{
    cout << "Not a valid dice throw" << endl;
}

if(dice == 1 || dice == 2 || dice == 3)
{
    cout << "You don't seem to have luck" << endl;
}

if(dice == 4 || dice == 5)
{
    cout << "Pretty good if you ask me" << endl;
}

if(dice == 6)
{
    cout << "Damn, you rolled a six. You didn't lie, did you ?" << endl;
}

return 0;
}
```

A terminal window titled 'cpp\_playground : bash — Konsole' showing the execution of a C++ program. The program prompts the user to 'Throw a dice and enter the result'. The user enters '4', and the program outputs 'Pretty good if you ask me'. The terminal prompt is '03:18:09 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp\_playground →' with a cursor.

```
cpp_playground : bash — Konsole
Throw a dice and enter the result 4
Pretty good if you ask me
03:18:09 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

```
cpp_playground : bash — Konsole
Throw a dice and enter the result -10
Not a valid dice throw
03:18:26 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

## Условен оператор if/else

Ако искаме да изпълним един код ако условието е изпълнено, и друг код ако условието не е изпълнено, вместо да използваме два `if`-а, може да използваме оператора `else`.

```
if(dice < 1 || dice > 6)
{
    cout << "Not a valid dice throw" << endl;
}
else
{
    cout << "You threw a valid dice, nice !" << endl;
}
```

Ако желаем да разгледаме повече от два случая (а не само когато дадено условие е изпълнено или не), може да използваме `else if`

```
if(dice == 1 || dice == 2 || dice == 3)
{
    cout << "You don't seem to have luck" << endl;
}
else if(dice == 4 || dice == 5)
{
    cout << "Pretty good if you ask me" << endl;
}
else
{
    cout << "Damn, you rolled a six. You didn't lie, did you ?" << endl;
}
```

Можем да влагаме `if` в други `if`-ове:

```
#include <iostream>
using namespace std;
```

```

int main()
{
    int dice = 1;

    cout << "Throw a dice and enter the result ";
    cin >> dice;

    if(dice < 1 || dice > 6)
    {
        cout << "Not a valid dice throw" << endl;
    }
    else
    {
        if(dice == 1 || dice == 2 || dice == 3)
        {
            cout << "You don't seem to have luck" << endl;
        }
        else if(dice == 4 || dice == 5)
        {
            cout << "Pretty good if you ask me" << endl;
        }
        else
        {
            cout << "Damn, you rolled a six. You didn't lie, did you ?" << endl;
        }
    }
    return 0;
}

```

## Област на видимост на променливите

Когато декларираме променлива, тя е видима (може да се достъпва) до края на текущия блок. Това се нарича scope на променливите, т.е. една променлива декларирана вътре в `if` няма да е видим извън него.

```

#include <iostream>
using namespace std;

int main()
{
    int condition = 3;
    if(condition > 1)
    {
        int result = 5;
    }

    cout << result;
    return 0;
}

```

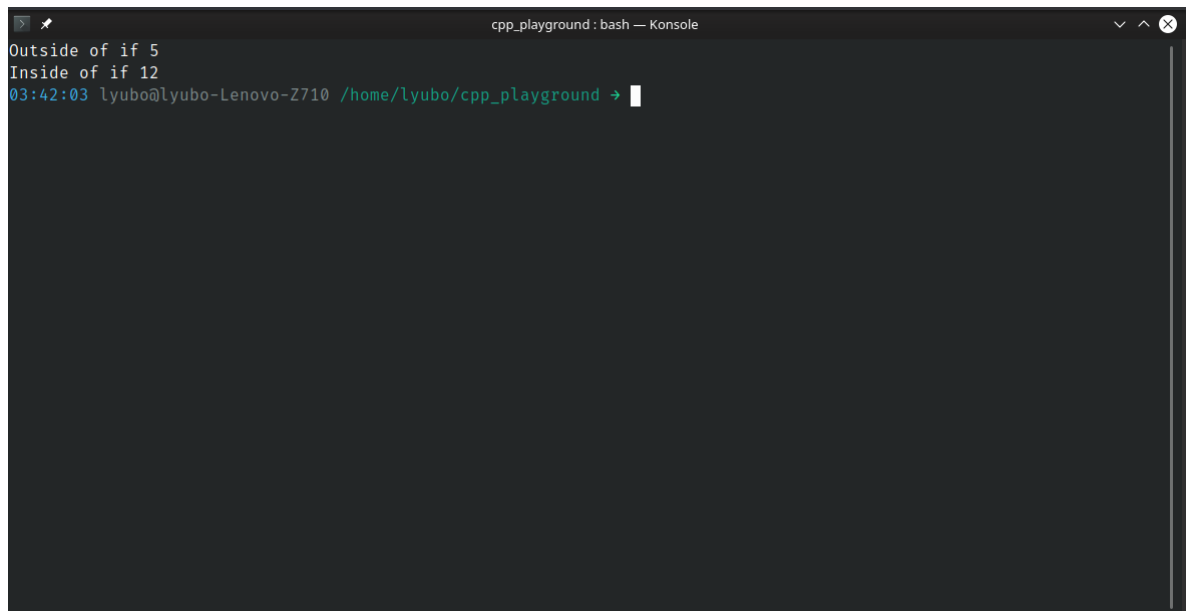
*Този код е невалиден*

В един блок не може да се декларират две променливи с едно и също име. Ако декларираме променлива със същото име в по-долен блок, то втората променлива ще замени първата до края на текущия блок.

```
#include <iostream>
using namespace std;

int main()
{
    int condition = 3;
    int test_variable = 5;
    cout << "Outside of if " << test_variable << endl;
    if(condition > 1)
    {
        int test_variable = 12;
        cout << "Inside of if " << test_variable << endl;
    }

    return 0;
}
```



```
cpp_playground : bash — Konsole
Outside of if 5
Inside of if 12
03:42:03 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

## Оператор switch

Оператора `switch` се използва най-често когато трябва да се сравни стойността на дадена променлива с набор от стойности. Нека разгледаме примера със зара отново:

```
#include <iostream>
using namespace std;

int main()
{
    int dice = 1;

    cout << "Throw a dice and enter the result ";
    cin >> dice;

    switch(dice)
    {
        case 1:
            cout << "You don't seem to have luck" << endl;
    }
}
```

```

        break;
    case 2:
        cout << "You don't seem to have luck" << endl;
        break;
    case 3:
        cout << "You don't seem to have luck" << endl;
        break;
    case 4:
        cout << "Pretty good if you ask me" << endl;
        break;
    case 5:
        cout << "Pretty good if you ask me" << endl;
        break;
    case 6:
        cout << "Damn, you rolled a six. You didn't lie, did you ?" << endl;
        break;
    default:
        cout << "Not a valid dice throw" << endl;
        break;
}
return 0;
}

```

Забелязваме наличието на нов оператор, оператора `break` - той служи за приключване на текущия блок. В тези случаи, блока съдържа всички `case`-ове. Поради тази причина, ако няма `break`, ще се изпълнят всички `case`-ове. `default` се изпълнява, ако стойността не се срещне в никой един `case`

## Тернарен оператор

Съществува оператор, който приема три аргумента - оператор `?:`

условие ? израз1 : израз2

Пример:

```
int a = (1 < 2) ? 100 : 200;
```

## Задачи:

1. Нека направим програма, която при въведени две числа от потребителя, да каже дали първото число е по-голямо или по-малко.
2. Напишете прост калкулатор, който да работи с цели числа - потребителя въвежда първото число, след това второто, след това знак, съответстващ на операцията, която иска да направи (+, -, \*, / или %). След въвеждане на тези три неща, програмата да изведе на екрана резултата
3. Променете калкулатора така, че да работи и с числа с плаваща запетая
4. Направете програма, в която потребителя въвежда три числа. Програмата има за цел да провери дали тези три числа са валидни дължини на страни на триъгълник.