**Умовний оператор if..else. Логічні операції.**

В повсякденному житті наші дії завжди залежать від якихсь умов. В залежності від чогось ми поступаємо так чи інакше. Так і в програмуванні є оператори для задання дій при якихось певних умов.

Перш ніж перейти до конкретики, давайте представимо як порядок наших дій залежить від певних умов. Приходячи додому ми всі відкриємо двері нашої домівки. Дома може нікого не бути і нам доведеться дістати ключі і відчинити двері. Або навпаки, хтось з членів сім’ї прийшов раніше і не зачинив двері, в такому випадку діставати ключ немає сенсу – двері уже відчинені. Якщо б довелось описати порядок дій в цій ситуації на мові програмування це виглядало б так:

If(двері\_зачинені){

Дістати ключ;

Відчинити замок;

Відчинити двері;

}

else{

Відчинити двері;

}

Але ми прекрасно знаємо, що в програмуванні ніяких дверей не має, про те є якісь умови на математичній основі. Давайте приведем більш конкретніший приклад реалізації умовного оператора, наприклад перевіримо чи має право особа водити машину. Ми знаємо що водійські права на машину можна отримати з 18 років. Давайте створимо умовний оператор, який буде перевіряти, чи можна особі в її віці отримати права. Створимо змінну «old», в якій визначимо вік, а потім перевіримо чи можна в цьому віці отримати водійські права на машину:

int old = 16;

if(old >= 18){

cout << “ More than the right age”;

}

Давайте розглянемо, що відбувається з точки зору програми. Ми присвоїли змінній «old» значення 16. Потім виконання програми переходить в оператор If. З оператором If поруч повинна завжди стояти умова, яка перевіряється на ІСТИНОСТЬ. В випадку, якщо умова істина - блок коду в фігурних дужках If(умова) {…} - БУДЕ виконуватись. Якщо умова не істина, то не буде виконуватись. В прикладі вище ми перевіряємо чи значення змінної «old» більше чи дорівнює 18. Так як значення змінної дорівнює 16, а 16 є меншим числом за 18, то результат умови не істина (false). Тому блок коду не був виконаний. Якщо ми встановимо змінній «old» значення 18 або більше, то порівняння в операторі If буде правильним, а це означає, що результат умови буде істинним (true). В умові оператора if ми можемо використовувати всі порівняльні операції в мові програмування С++, а саме:

більше: old > 18,

менше: old < 18,

більше чи дорівнює: old >= 18,

менше чи дорівнює: old <= 18,

чи дорівнює: old == 18,

чи не дорівнює: old != 18

Наведемо приклад з цими умовами:

#include <iostream>

using namespace std;

int main()

{

int old = 18;

if(old < 18){

cout<<"old < 18\n";

}

if(old > 18){

cout<<"old > 18\n";

}

if(old <= 18){

cout<<"old <= 18\n";

}

if(old >= 18){

cout<<"old >= 18\n";

}

if(old == 18){

cout<<"old == 18\n";

}

if(old != 18){

cout<<"old != 18\n";

}

return 0;

}

Програма виведе повідомлення:

old <= 18

old >= 18

old == 18

Там де умова істина блок коду виконався, там де ні – його було пропущено.

За необхідності можна використовувати повну форму оператору if – else. Else сам по собі не існує, а тільки в парі з if. Якщо умова в if не виконується, то буде виконаний блок else, якщо він є в наявності. Один else відповідає одному if. Перепишемо код:

#include <iostream>

using namespace std;

int main()

{

int old = 18;

if(old < 18){

cout<<"old < 18 - true\n";

}

else{

cout<<"old < 18 - false\n";

}

if(old > 18){

cout<<"old > 18 - true\n";

}

else{

cout<<"old > 18 - false\n";

}

if(old <= 18){

cout<<"old <= 18 - true\n";

}

else{

cout<<"old <= 18 - false\n";

}

if(old >= 18){

cout<<"old >= 18 - true\n";

}

else{

cout<<"old >= 18 - false\n";

}

if(old == 18){

cout<<"old == 18 - true\n";

}

else{

cout<<"old == 18 - false\n";

}

if(old != 18){

cout<<"old != 18\n";

}

return 0;

}

Результат відпрацювання:

old < 18 - false

old > 18 - false

old <= 18 - true

old >= 18 - true

old == 18 - true

Це нам говорить те, що там де умова не виконалась – виконався блок else. В останньому if умова false, блока else немає, тому нічого не відбулось.

Також можливо робити більш складніші умови в операторі if. Поєднати умови можливо за допомогою логічних операцій І(and) - && , АБО(or) - ||. Операція and істинна, якщо всі об’єднані нею умови істинні. Операція or істина, якщо хоча б одна умова істина. Наприклад вік більше за 18 І зріст вище 180. Вік більше 18 АБО зріст вище 180. Прилад операції and:

if((old > 18)&&(height > 180)){

cout<<"Він потрапив сюди, тому що йому більше 18 за віком і він вище 180см";

}

else{

cout<<"Йому не вистачило або віку або росту";

}

Прилад операції or:

if((old > 18)||(height > 180)){

cout<<"Він потрапив сюди, тому що йому більше 18 за віком або він просто дуже високий";

}

else{

cout<<" Йому не вистачило ні віку ні росту ";

}

Розглянемо більш математичний приклад. Наприклад, нам потрібно перевірити, чи значення "х" додатне. Використаємо для цього конструкцію оператора if..else

#include <iostream>

using namespace std;

int main()

{

int х = 3;

if(x > 0){

cout<<"Додатній Х";

}

else{

cout<<"Відємний Х";

}

х = -5;

if(x > 0){

cout<<"Додатній Х";

}

else{

cout<<"Відємний Х";

}

return 0;

}

В цьому прикладі спочатку виведеться повідомлення що "х" додатній, тому що його значення дорівнює 3, і в операторі if умова виконується, так як 3 > 0. Потім виведеться повідомлення, що "х" від’ємний, тому що ми присвоїли "х" нове значення -5. В операторі if умова -5 > 0 не виконується, так як -5 менше 0, тому виконується блок else.

Таким чином ми можемо перевіряти значення "х" на певну умову. Розглянемо ще один приклад, де потрібно повідомити, якщо х лежить в межах від -3 включно до 0. Для цього використаємо логічну операцію and щоб об’єднати умови:

#include <iostream>

using namespace std;

int main()

{

int х = 3;

if( (x >= -3) && (x < 0)){

cout<<"Х входить в діапазон";

}

else{

cout<<"Х не входить в діапазон";

}

х = -2;

if( (x >= -3) && (x < 0)){

cout<<"Х входить в діапазон";

}

else{

cout<<"Х не входить в діапазон";

}

return 0;

}

Як вже було сказано вище логічний оператор and(&&) повертає істину, якщо дві умови істині. Даний приклад виведе спочатку, що "х" не входить в діапазон, тому що друга умова не виконується (x < 0), хоча перша і виконується (x >= -3). Слідом буде виведено, що "х" входить в діапазон. Ми присвоїли "х" нове значення -2. З таким значення обидві умови (x >= -3 та x < 0) дійсні, тому логічний оператор and(&&) поверне істину, і буде виконаний блок коду if.

Розглянемо ще один приклад, не на прикладі діапазону, а на прикладі допустимих значень. Наприклад в нас є умова, що х лежить в межах від мінус нескінченності до 0, або від 10 включно до плюс нескінченності. Давайте подивимось як перевірити "х", чи відповідає його значення значенню в цих величинах.

#include <iostream>

using namespace std;

int main()

{

int х = 3;

if( (x < 0) || (x >= 10)){

cout<<"Х або менше 0, або быльше-дорівнює 10";

}

else{

cout<<"Х лежить в діапазоні від 0 до 10";

}

х = -2;

if( (x < 0) || (x >= 10)){

cout<<"Х або менше 0, або быльше-дорівнює 10";

}

else{

cout<<"Х лежить в діапазоні від 0 до 10";

}

return 0;

}

Як вже було сказано вище логічний оператор or(||) повертає істину, якщо хоча б одна умова істина. Даний приклад виведе спочатку, що "х" лежить в діапазоні від 0 до 10, тому що друга ні одна умова не виконується 3 не менше 0 ( 3 < 0) і 3 не більше-рівне 10 (3 >=10). Так як ні одна умова не істина результат оператора or(||) буде не істинним. Слідом буде виведено, що "х" або менше 0, або більше-дорівнює 10 . Ми присвоїли "х" нове значення -2. З таким значення одна умова (x < 0 ) виконується, тому логічний оператор or(||) поверне істину, і буде виконаний блок коду if.