

Loops and ifs

Author Andrew Lycas

Ok, i lied, this isn't going to be the calculator, that will be next. I had forgotten that you needed one more thing before you can do the calculator, and that is loops and ifs.

If statements are easier so we will start with those.

```
--  
if (x == y)  
{  
    //do this  
}  
--
```

above is the basic syntax for an if statement. Translated it is saying, if x is equal to y, than do what ever is in the pointy brackets. Notice the two equal signs. One equal sign would mean that you are setting the value of the variable on the left, but when there are two it is checking to make sure that both are equal to each other. Other operates include:

```
== equal  
!= not equal  
> greater than  
< less than  
>= greater than or equal to  
<= less than or equal to
```

You can use any of those within the parenthesis in the if statement.

With if statements you can also use else statements and else if statements.

```
--  
if (x == y)  
{  
    //do this  
}  
else  
{  
    // or do this  
}  
--
```

if x is equal to y than do something, but if it isn't do something else. Notice how the else is right under the if statement. With an else it will only do it if the if statement is not satisfied.

```

--
if (x != y)
{
    // do this
}
else if(x == 2)
{
    // do this only if x and y are equal and x is equal to 2
}
--

```

now you can have as many else ifs as you want

```

--
#include <midstream>
#include "stdio.h"

int main()
{

    using namespace std; // just makes it so i do not have to keep typing std::cout
                          // i can just leave out the std:: part

    int x = 2;
    int t = 5;

    if (x==t)
    {
        cout << "test" ;
    }
    else if(x== 4)
    {
        cout << "test2" ;
    }
    else if (x==9)
    {
        cout << "test3";
    }
    else
    {
        cout << "test4";
    }
    getchar();
}
--

```

above it should print test 4.

if you have any questions about if / else / else if statements, just ask them.

Another type of conditional, or loop would be the while statement.

```
--  
x = 3  
  
while (x == 3)  
{  
    //do this until x is no longer equal to three  
}  
--
```

above is an example of a while loop. It will do what ever it is told until the condition in the () is not true.

```
--  
  
#include <iostream>  
#include "stdio.h"  
  
int main()  
{  
  
    using namespace std;  
  
    int x = 1;  
  
    while (x < 10)  
    {  
        cout << x ;  
        x++; // this is the same as x=x+1 , it is just sort of a short hand  
            // other short hand includes x+=5 would be x = x+5  
    }  
  
    getchar();  
}  
--
```

put that into your complier and see what the output is. It should be 123456789 .

one thing to remember about while loops is that sometimes you can get what is called an infinite loop, which is where the while statement's condition (what is in the ()) is never not true.

Something else to remember is that with an if statement it will only check when it gets to the end of the while statement to see if the condition is true.

```

--
x = y;
while (x=y)
{
    x++; // now x is not equal to y, but the while statement will still continue to go through till the
        //end of itself
    y++ ; // now x and y are equal again.
}
// the program will never get through the while statement causing an infinite loop.
--

```

the third type of loop statement that i will be teaching (i wont be teaching them all, some are rarely used, do some research..) is the for loop.

The for loop has three parts.

```

for( int i ; i < 10 ; i++)
{
    cout << "round" << i << "\n";
}

```

inside the parenthesis the first of the 3 parts, is the declaration of a variable This variable will be used for the for statement to keep track of what number it is on. This variable can be used within the for statement, but you will not be able to u what ever number is used in that variable for the rest of the program. The second part is just saying do the stuff inside until this condition is not true anymore. And the third part is telling how to change the variable each round.

So this, if put into a program would print out :

```

round 0
round 1
round 2
round 3
round 4
round 5
round 6
round 7
round 8
round 9

```

for statements are used when u know how many times something needs to be done.

For example if u needed something to be done 5 times:

```

--
for (int i; i< 5 ; i++)
{
}
--

```

look back to the first set of for code i gave you. Notice how i just said int i, and the first number was a 0. if you give no value to the computer for a variable, but u declare said variable It will be given the default value of 0. In the second for example, notice how in the second part, it is just < rather than <=. if it were <= then the loop would go through 6 times, because i started at 0.

```
--  
for (int i; i< 5 ; i++)  
{  
}  
  
for (int i=1; i<= 5 ; i++)  
{  
}  
--
```

both of those do the same thing, use whichever one you are most comfortable with. I usually use the first one, but that is just a personal preference

Alright, those are the three types of loops/conditional statements that i use the most, there are a couple more(case statements, and do while statements) the case statement i will show u in the intro to the next lesson and the do while statements are kinda pointless in my opinion, but just look them up if u are curious..

next time: project 1: the calculator.