Loop Instructions

IN 1101 PROGRAMMING FUNDAMENTALS

Loops

- Loop control instructions are used to perform set of instructions repeatedly.
- Involves repeating certain instructions number of times until a particular condition is being satisfied.
- Methods by which we can repeat:
  - Using **for** statements.
  - Using **while** statements.
  - Using **do-while** statements.

while Loop

- Used when you need to repeat something a fixed number of times.
- General form:

```
initialize loop counter ;
while ( test loop counter using a condition )
{
    do this ;
    and this ;
    increment loop counter ;
}
```

while Loop Cont.

- The condition being tested may use relational or logical operators.

e.g.:

```
while ( i <= 10 )
while ( i >= 10 && j <= 15 )
while ( j > 10 && ( b < 15 || c < 20 ) )
```

Operator Precedence Revisit

Refer: https://www.cs.uic.edu/~i109/Notes/COperatorPrecedenceTable.pdf

Example

- Write a program to print the numbers from 1 to 10.

Is This Correct?

```
# include <stdio.h>
int main( )
{
    int i = 1 ;
    while ( i <= 10 )
        printf ( "%d\n", i ) ;
    return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
    int i = 1 ;
    while ( i <= 10 )
    {
        printf ( "%d\n", i ) ;
        i++ ;
    }
    return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
    int i = 1 ;
    while ( i <= 10 )
    {
        printf ( "%d\n", i ) ;
        i += 1 ;
    }
    return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
    int i = 0 ;
    while ( i++ < 10 )
        printf ( "%d\n", i ) ;
    return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
    int i = 0 ;
    while ( ++i <= 10 )
        printf ( "%d\n", i ) ;
    return 0 ;
}
```

Try This !

Write a program to find the factorial value of any number entered through the keyboard.

for Loops

- for allows to specify three things about the loop.
  - Initial value of the loop counter
  - Condition to test the counter to determine whether to continue with for loop.
  - Increment/Decrement counter.
- General form:

```
for ( initialize counter ; test counter ; increment counter )
{
    do this ;
    and this ;
}
```

Example

Re-write the program to print the numbers from 1 to 10 using for loop.

```
# include <stdio.h>
int main( )
{
    int i ;
    for ( i = 1 ; i <= 10 ; i = i + 1 )
        printf ( "%d\n", i ) ;
    return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
      int i ;
      for ( i = 1 ; i <= 10 ; )
      {
            printf ( "%d\n", i ) ;
            i = i + 1 ;
       }
       return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
    int i = 1 ;
    for ( ; i <= 10 ; i = i + 1 )
        printf ( "%d\n", i ) ;
    return 0 ;
}
```

What about this?

```
# include <stdio.h>
int main( )
{
    int i = 1 ;
    for ( ; i <= 10 ; )
    {
        printf ( "%d\n", i ) ;
        i = i + 1 ;
    }
    return 0 ;
}
```

Nesting Loops

Write the output of the following program.

```
# include <stdio.h>
int main( )
{
    int r, c, sum ;
    for ( r = 1 ; r <= 3 ; r++ ) /* outer loop */
    {
        for ( c = 1 ; c <= 2 ; c++ ) /* inner loop */
        {
            sum = r + c ;
            printf ( "r = %d c = %d sum = %d\n", r, c, sum ) ;
        }
    }
    return 0 ;
}
```

Try This !

Write a program to display a rectangular start pattern for given two different input numbers.

E.g Sample input numbers : 3,4

Sample output : ***

***

***

***

break and *continue* Statements

- ***break*** is used to jump out of a loop instantly, without waiting to get back to the condition.
- ***continue*** allows to take the control to the beginning of the loop, bypassing the statements inside the loop, which have not yet been executed.

What is the Output?

```
#include<stdio.h>
void main ()
{
    int i;
    for(i = 0; i<10; i++)
    {
        printf("%d ",i);
        if(i == 5)
            break;
    }
    printf("came outside of loop at i = %d",i);
}
```

What is the Output?

```
#include<stdio.h>
void main ()
{
    int i;
    for(i = 0; i<10; i++)
    {
        if(i == 5)
            continue;
        printf("%d ",i);
    }
}
```

do-while Loop

- ***do-while*** test the condition after having executed the statements within the loop.
- Executes at least once.

```
do
{
    this ;
    and this ;
    and this ;
    and this ;
} while ( this condition is true ) ;
```

Questions?