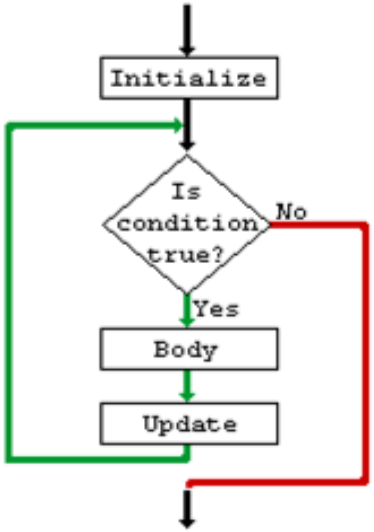
Loop Control Structures

L9-T4



Learning Objectives

- To learn and appreciate the following concepts
  - break
  - continue
  - typedef and enum

Learning Outcome

At the end of session the student will be able to

- break
- continue
- typedef and enum

The break Statement

- Used in order to immediately exit from a loop
- After a break, following statements in the loop body are skipped and execution continues with the first statement after the loop
- If a break is executed from within nested loops, only the innermost loop is terminated

Exiting a loop with break statement

```
while (…….)
{…….
…….
If(condition)
   break;
……..
……..
} // end of while
…….. //next
          statement
```

Exit From loop

```
do
{…….
…….
If(condition)
   break;
……..
……..
} while(…);
…….. // next
          statement
```

Exit From loop

Break Statement Examples: Check whether given number is prime or not

```
   int   j=2,  prime=1;
    scanf("%d",&N);
while(j<N)
 {
     if( (N % j) == 0)
     {
      prime=0;
      break; /* break out of for loop */
     }
J++;
  }
  if (prime == 1)
       printf("%d is a prime no",N);
else
       printf("%d is a not a prime no",N);
```

Program to generate prime numbers between given 2 limits

```
scanf("%d %d",&m,&n);
i=m;
while(i<n)
{  int prime=1, j=2;
while(j<i)
    {
        if( i % j == 0)
        {
            prime=0;
            break; /* break out of inner loop */
        }
            J++;
     }
if (prime == 1) printf("%d\t",i);
i++;
}
```

Skipping a part of loop-continue statement

- Skip a part of the body of the loop under certain conditions is done using **continue** statement.

- As the name implies, **continue** causes the loop to be continued with next iteration, after skipping rest of the body of the loop.

```
while (.....)
{
   Statement-1;
   Statement-2;

  If(condition)
        continue;

   Statement-3;
   Statement-4;
}
Next_statement
```

```
do
{
   Statement-1;
   Statement-2;

   If(condition)
        continue;

   Statement-3;
   Statement-4;
} while(...);

Next_statement
```

Continue Statement

```
#include<stdio.h>
int main()
{   int j=0;
   while( j<=8)
   {
      if (j==4)
    { j++ ;
     continue;
   }
   printf("%d ", j);
    j++;
   }
   return 0;
}
```

Output: 0 1 2 3 5 6 7 8

```
#include<stdio.h>
int main()
{   int j=0;
   while( j<=8)
   {
      if (j==4)
    {
     continue;
   }
   printf("%d ", j);
    j++;
   }
   return 0;
}
```

Output:0 1 2 3

User defined Type declarations

- **typedef**
  - **Type definition – lets you define your own identifiers.**
- **enum**
  - **Enumerated data type - a type with restricted set of values.**

User defined Type Declaration

- *typedef* **type identifier;**

  The "type" refers to an existing data type and "identifier" refers to the new name given to the data type.

- After the declaration as follows:

  typedef **int marks;**

  typedef **float units;**

  we can use these to declare variables as shown

  marks *m1,m2* **;** //*m1* & *m2*are declared as integer variables

  units *u1*, *u2***;** //*u1* & *u2* are declared as floating point variables

  The main advantage of typedef is that we can create meaningful data type names for increasing the readability of the program.

User defined Type Declaration - enum

enum identifier { value1, value2,..,value$_n$ };

- Here, ***identifier*** is the name of enumerated data type or tag. And ***value1, value2,....,valueN*** are values of type identifier.

- By default, *value1* will be equal to 0, *value2* will be 1 and so on but, the programmer can change the default value.

```
enum card {club, diamonds, hearts, spades};

enum card {club=0, diamonds=10, hearts=20, spades=3};
```

Algorithm and Program for Fibonacci series

Algorithm : Fibonocci Series
Step 1 : Input Limit
Step 2: First←0,Second←1
Step 3: print First
Step 4: WHILE Second < Limit
begin
Print Second
Next← First + Second
First←Second
Second← Next
end
Step 5:[End of Algorithm]
Stop

```
#include<stdio.h>
int main()
{
    int first=0, second=1;
   int limit, next;

   scanf("%d",&limit);
    printf("%d",first);

   while(second < limit)
   {
        printf("%d",second);
       next = first + second;
       first = second;
        second = next;
    }
  return 0;
}
```

Tutorial Problems

- **Write a C program to print all natural numbers from n-0 in reverse using while loop**
- **Write a C program to find last and first digit of any number**
- **Write a C program to enter any number and print all its factors**
- **Write a C program to find LCM of two numbers**
- **Write a C program to convert Binary to Octal number**

Session 8 Summary

- The `do` Loop
- The `break` Statement
- The `continue` Statement
- Typedef and Enum

Poll Question

Go to chat box/posts for the link to the Poll question

Submit your solution in next 2 minutes

Click the result button to view your score