CS1100 – Introduction to Programming

Lecture 9

Instructor: Shweta Agrawal (shweta.a@cse.iitm.ac.in)

for construct

- Syntax

```
for (stmt1; expr; stmt2) {
        statements;
}
```

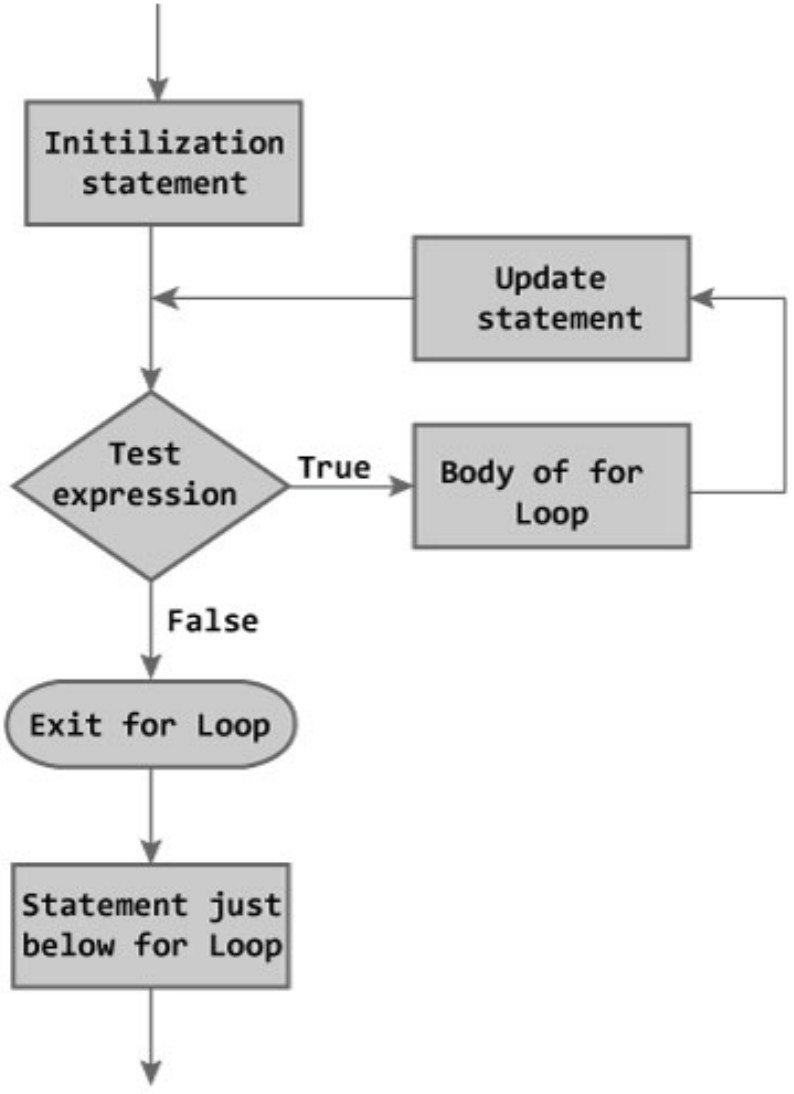
for construct

- Syntax

  for (stmt1; expr; stmt2) {
  statements;
  }

- Semantics
  1. Execute stmt1.
  2. If expr is true, execute statements. execute stmt2. goto step 2.
  3. If expr is false, exit loop.

- expr must be changed to ensure that it is not an infinite loop.



Sum of 100 integers – using for loop

User enters 100 integers. Sum them and print the sum.

Sum of 100 integers – using for loop

User enters 100 integers. Sum them and print the sum.

```
#include<stdio.h>
int main() {
    int input;
    int count;
    int sum = 0;

    for (count=1; count <= 100; count++) {
         printf("Enter an integer: \t");
         scanf(" %d", &input);
         sum +=  input;
    }
    printf("sum of numbers entered is %d\n", sum);
    return 0;
}
```

Typically, stmt1 : initialization and stmt2 : increment.

do while construct

- Syntax

  do

  { statements; }

  while (expr);

do while construct

- Syntax

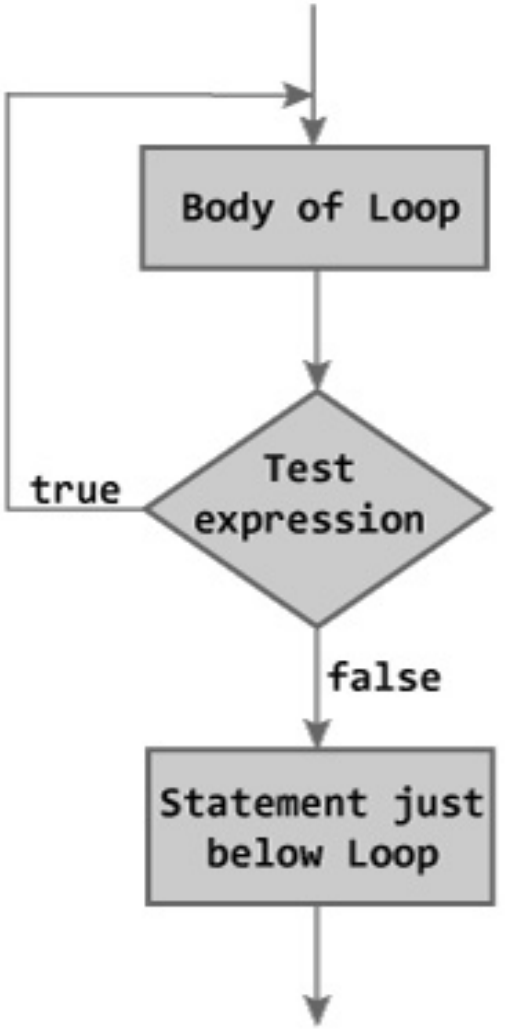
  do

  { statements; }

  while (expr);

- Semantics
  1. Execute statement.
  2. If expr is true, goto step 1, else exit loop.

- expr must be changed by statement.



Sum of 100 integers – using do while loop

User enters 100 integers. Sum them and print the sum.

```
#include<stdio.h>
int main() {
    int input;

    int count = 1;
    int sum = 0;

    do {
         printf("Enter an integer: \t");
         scanf(" %d", &input);
         count++;
         sum +=  input;
    } while (____);

    printf("sum of numbers entered is %d\n", sum);
    return 0;
}
```

Sum of 100 integers – using do while loop

User enters 100 integers. Sum them and print the sum.

```
#include<stdio.h>
int main() {
    int input;

    int count = 1;
    int sum = 0;

    do {
         printf("Enter an integer: \t");
         scanf(" %d", &input);
         count++;
         sum +=  input;
    } while (count <= 100);

    printf("sum of numbers entered is %d\n", sum);
    return 0;
}
```

Sum of 100 integers – using do while loop

User enters 100 integers. Sum them and print the sum.

```
#include<stdio.h>
int main() {
    int input;

    int count = 1;
    int sum = 0;

    do {
         printf("Enter an integer: \t");
         scanf(" %d", &input);
         count++;
         sum +=  input;
    } while (count <= 100);

    printf("sum of numbers entered is %d\n", sum);
    return 0;
}
```

Sum of 100 integers – using do while loop

User enters either 100 integers or terminates the program by entering -1. Sum the integers input and print the sum.

Sum of 100 integers – using do while loop

User enters either 100 integers or terminates the program by entering -1. Sum the integers input and print the sum.

```
#include<stdio.h>
main() {
int     int input;

    int count = 1;
    int sum = 0;

    do {
         printf("Enter an integer: \t");
         scanf(" %d", &input);
         count++;
         if (input != -1) sum +=  input;
    } while (count <= 100 && input != -1);

    printf("sum of numbers entered is %d\n", sum);
    return 0;
}
```

`break` statement in loops in C

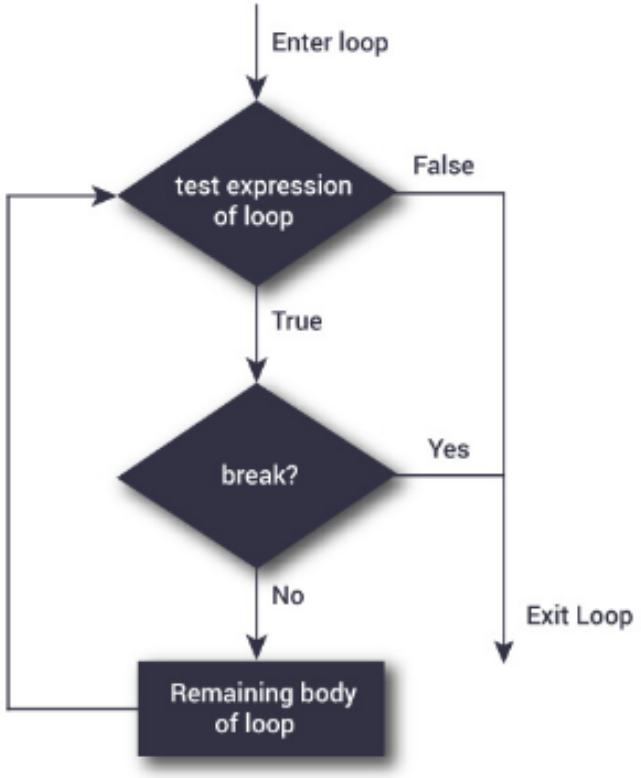
`break` : break out of the loop execution completely.

`break` statement in loops in C

`break` : break out of the loop execution completely.

```
while (test Expression)
{
    // codes
    if (condition for break)
    {
        break;
    }
    // codes
}
```

```
for (init, condition, update)
{
    // codes
    if (condition for break)
    {
        break;
    }
    // codes
}
```



Example : using `break`

User enters either 100 integers or terminates the program by entering -1. Sum the integers input and print the sum.

Example : using break

User enters either 100 integers or terminates the program by entering -1. Sum the integers input and print the sum.

```
#include <stdio.h>
int main() {
    int i;
    int number, sum = 0;

    for(i=1; i <= 100; i++)
    {
        printf("Enter n%d: ",i);
        scanf("%d",&number);

        /* If user enters negative number, loop is terminated */
        if (number == -1) break;

        sum = sum + number;
    }
    printf("Sum = %d",sum);
    return 0;
}
```

`continue` statement in loops in C

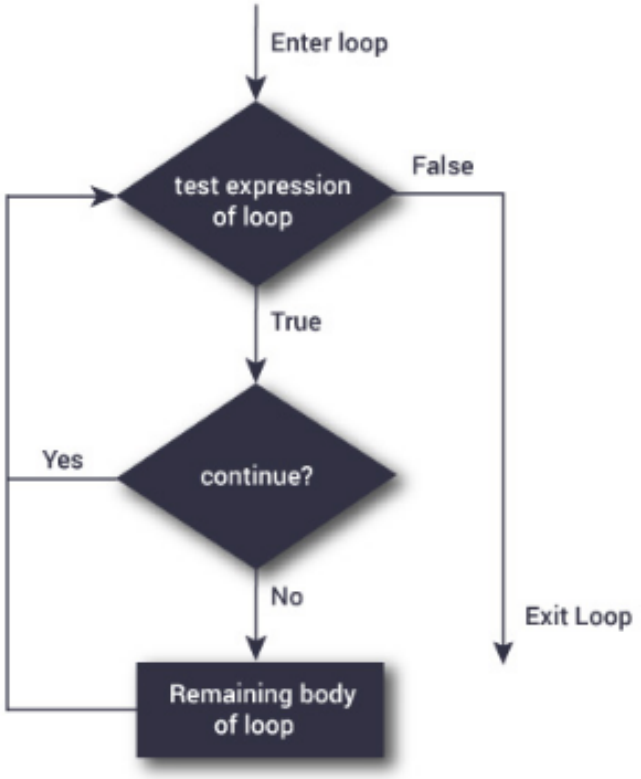
`continue` : skip the rest of iteration and go to next iteration.

`continue` statement in loops in C

`continue` : skip the rest of iteration and go to next iteration.

```c
while (test Expression)
{
    // codes
    if (condition for continue)
    {
        continue;
    }
    // codes
}
```

```c
for (init, condition, update)
{
    // codes
    if (condition for continue)
    {
        continue;
    }
    // codes
}
```



Example : using `continue`

User enters either 100 integers some of which may be negative.
Sum the positive integers given and print the sum.

Example : using continue

User enters either 100 integers some of which may be negative.
Sum the positive integers given and print the sum.

```
#include <stdio.h>
int main() {
    int i;
    int number, sum = 0;

    for(i=1; i <= 100; i++)
    {
        printf("Enter n%d: ",i);
        scanf("%d",&number);

        /* If user enters negative number, it is skipped from sum */
        if (number < 0) continue;

        sum = sum + number;
    }
    printf("Sum = %d",sum);
    return 0;
}
```

Different Loop constructs - and related ...

- **while, for, do while.**
- Why have different constructs?
- Can you replace one by another?
  Yes, with careful modifications
- Each of the construct is natural for different problems.
- `break`, `continue` to handle control flow within the runs of the loop.

Computing 2^n

Accept n from the user. Print 2^n on the standard output.

```
#include<stdio.h>

int main() {
    int n;
    int value = 2;

    printf("Enter a positive integer:\t");
    scanf("%d", &n);

    while (____) {
       value *= 2;
       ______;
    }
    printf("%d\n", value);
    return 0;
}
```

Computing 2^n

Accept n from the user. Print 2^n on the standard output.

Computing 2^n

Accept n from the user. Print 2^n on the standard output.

```
#include<stdio.h>

int main() {
    int n;
    int value = 2;

    printf("Enter a positive integer:\t");
    scanf("%d", &n);

    while (n > 1) {
       value *= 2;
       n--;
    }
    printf("%d\n", value);
    return 0;
}
```

More examples

- Enter a Number and Check Whether that Number is a Perfect Number or Not. If the sum of all factors equals that number, it is called a perfect number.

More examples

- Enter a Number and Check Whether that Number is a Perfect Number or Not. If the sum of all factors equals that number, it is called a perfect number.
- Write a program in C to find the prime numbers within a range of numbers.

More examples

- Enter a Number and Check Whether that Number is a Perfect Number or Not. If the sum of all factors equals that number, it is called a perfect number.
- Write a program in C to find the prime numbers within a range of numbers.
- Write a C program to check whether a number is a palindrome or not.