CS1100 – Introduction to Programming

Lecture 8

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

- Programming : From Turtle to C.
- Data Types in C, Representations, Operators.
- Formatting the Input and the Output.
- Execution of Programs, Compilers.
- Modifying the control flow in Programs `if-then-else`, `switch`.

} So Far

CS1100 – Introduction to Programming

Lecture 8

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

- Programming : From Turtle to C.
- Data Types in C, Representations, Operators.
- Formatting the Input and the Output.
- Execution of Programs, Compilers.
- Modifying the control flow in Programs `if-then-else`, `switch`.

} So Far

- `while`, `for`, `do while` constructs in C.
- Example problems.
- Programming for engineers.

} Up Next

The **while** construct

- Syntax

  while (expression) {

  statements;

  }

The **while** construct

- Syntax

  while (expression) {

  statements;

  }

- Semantics

  1. As long as expression is true, execute statements.
  2. If expression is false, exit the loop.

The **while** construct

- Syntax

  while (expression) {

  statements;

  }

- Semantics
  1. As long as expression is true, execute statements.
  2. If expression is false, exit the loop.
- Value of expression must be changed by the body of the loop, otherwise we have an infinite loop.

The **while** construct

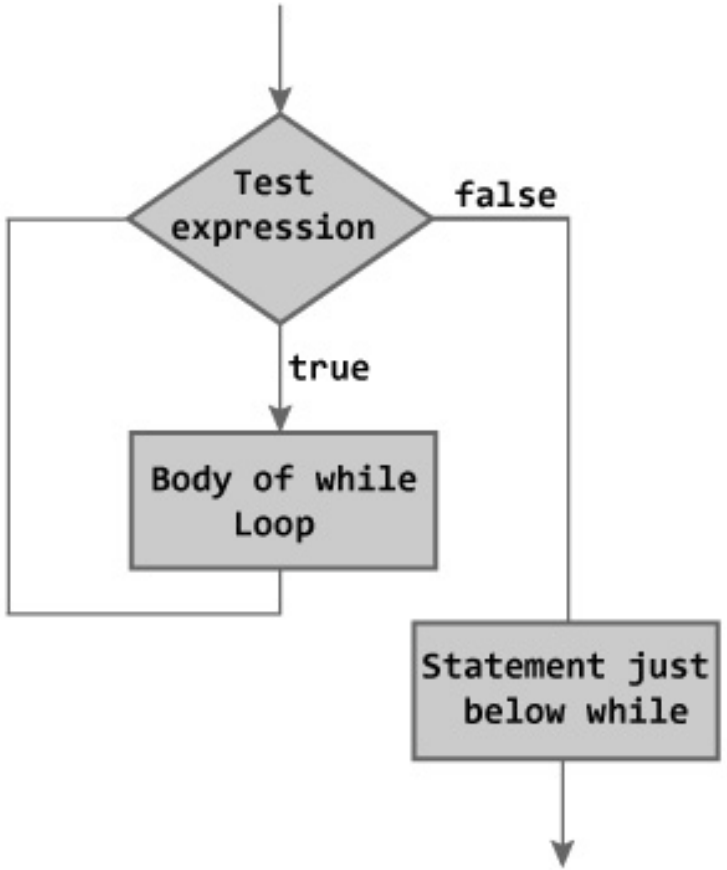
- Syntax

  while (expression) {

  statements;

  }
- Semantics
  1. As long as expression is true, execute statements.
  2. If expression is false, exit the loop.
- Value of expression must be changed by the body of the loop, otherwise we have an infinite loop.
- expression can contain relational, logical or equality operators.

Relational	<= < > >=
Equality	== !=
Logical	&& ||



Reversing the digits of a given unsigned integer

```
#include "stdio.h"
int main () {
  int number, revNumber, remainder;
  revNumber = 0;
  printf ("Input number:");
  scanf ("%d", &number);
  while (number > 0) {
    remainder = number % 10;
    revNumber = revNumber*10 + remainder;
    number = number/10;
  }
  printf ("The reversed number is : %d\n", revNumber);
}
```

Example: Sum even and odd numbers

Accept integers from the standard input as long as the user does not enter -1. Once the user enters -1, print the sum of all integers entered so far, sum of even integers and sum of odd integers.

Two useful constructs:

- while loop — repetitive statement
- switch — multiple selection

Summing up odd and even numbers

Is the program correct?

```
#include<stdio.h>
int main() {
    int input;
    int sum, eSum, oSum;
    printf("Enter an integer: \t");
    scanf(" %d", &input);

    while (input != -1) {
        sum += input;
        switch (input % 2) {
            case 0: eSum += input; break;
            case 1: oSum += input;
        }
    }
    printf("sum = %d, oddSum = %d, evenSum = %d\n", sum, oSum, eSum);
    return 0;
}
```

Summing up odd and even numbers

Is the program correct?

```
#include<stdio.h>
int main() {
    int input;
    int sum, eSum, oSum;
    printf("Enter an integer: \t");
    scanf(" %d", &input);

    while (input != -1) {
        sum += input;
        switch (input % 2) {
            case 0: eSum += input; break;
            case 1: oSum += input;
        }
    }
    printf("sum = %d, oddSum = %d, evenSum = %d\n", sum, oSum, eSum);
    return 0;
}
```

- common mistake: forgotten initialization.
- expr. not modified in body of loop.

Summing up odd and even numbers

```
#include<stdio.h>
int main() {
    int input;
    int sum, eSum, oSum;
    printf("Enter an integer: \t");
    scanf(" %d", &input);
    sum = eSum = oSum = 0; // initialization.

    while (input != -1) {
        sum += input;
        switch (input % 2) {
            case 0: eSum += input; break;
            case 1: oSum += input;
        }
        printf("Enter an integer: \t");
        scanf(" %d", &input);
    }
    printf("sum = %d, oddSum = %d, evenSum = %d\n", sum, oSum, eSum);
    return 0;
}
```

What does this code do?

```
#include <stdio.h>
int main() {
    int count = 0;
    while (count < 10) {
        if (count == 5) {
            break;
         }
        printf("Count is: %d\n", count);
        count++;
    }
    return 0;
}
```

What does this code do?

```
#include <stdio.h>
int main() {
    int count = 0;
    while (count < 5) {
        count++;
        if (count == 3) {
            continue; // Skip the rest of the loop body for
        }
        printf("Count is: %d\n", count);
    }
    return 0;
}
```

Examples

- Enter a number from the keyboard and then calculate the number of digits and the sum of digits of that number using a while loop.

Examples

- Enter a number from the keyboard and then calculate the number of digits and the sum of digits of that number using a while loop.
- Enter a number and print the Hemchandra/Fibonacci series up to that number using a while loop.

Examples

- Enter a number from the keyboard and then calculate the number of digits and the sum of digits of that number using a while loop.
- Enter a number and print the Hemchandra/Fibonacci series up to that number using a while loop.
- Read a set of n numbers (n is input) and print if each given number is smaller or bigger than the previous number. For first number there will not be any output as there is no previous number.