CS1100 – Introduction to Programming

Lecture 7

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

Goals:

- Selection statements:
  - Single Selection : if
  - Double Selection : if else
  - Multiple Selection: switch

Goals:

- Selection statements:
  - Single Selection : if
  - Double Selection : if else
  - Multiple Selection: switch
- Loops:
  - while
  - for
  - do while

Goals:

- Selection statements:
  - Single Selection : if
  - Double Selection : if else
  - Multiple Selection: switch
- Loops:
  - while
  - for
  - do while
- Need for different kinds of selection and loops.

Goals:

- Selection statements:
  - Single Selection : if
  - Double Selection : if else
  - Multiple Selection: switch
- Loops:
  - while
  - for
  - do while
- Need for different kinds of selection and loops.
- Control flow for each of the constructs.

Single Selection : `if` construct

Decide to execute a part of the program if a condition is true.

Single Selection : `if` construct

Decide to execute a part of the program if a condition is true.

Eg : If a number (say meant to represent marks) is negative print a warning.

Single Selection : `if` construct

Decide to execute a part of the program if a condition is true.

Eg : If a number (say meant to represent marks) is negative print a warning.

Syntax :

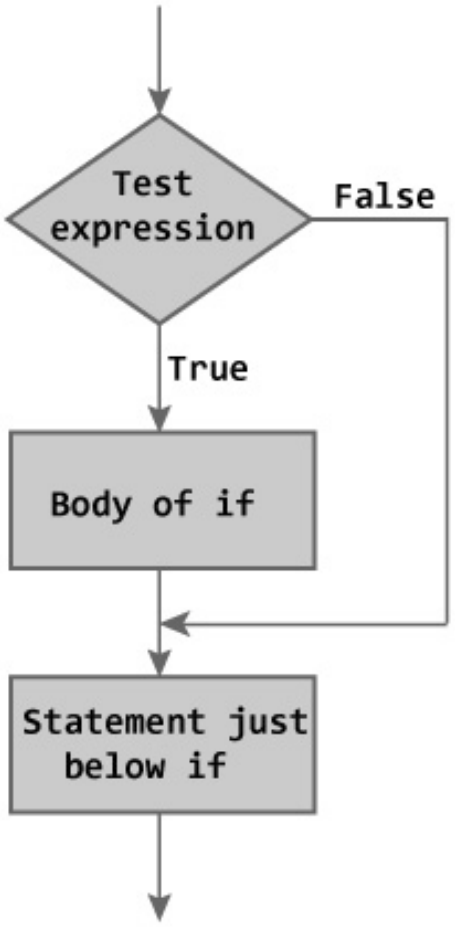
```
if (condition)
{ Statement Sequence 1 }
```

Single Selection : `if` construct

Semantics (meaning) :

Decide to execute a part of the program if a condition is true.

Eg : If a number (say meant to represent marks) is negative print a warning.



Syntax :

```
if (condition)
{ Statement Sequence 1 }
```

Single Selection : `if` construct

Example :

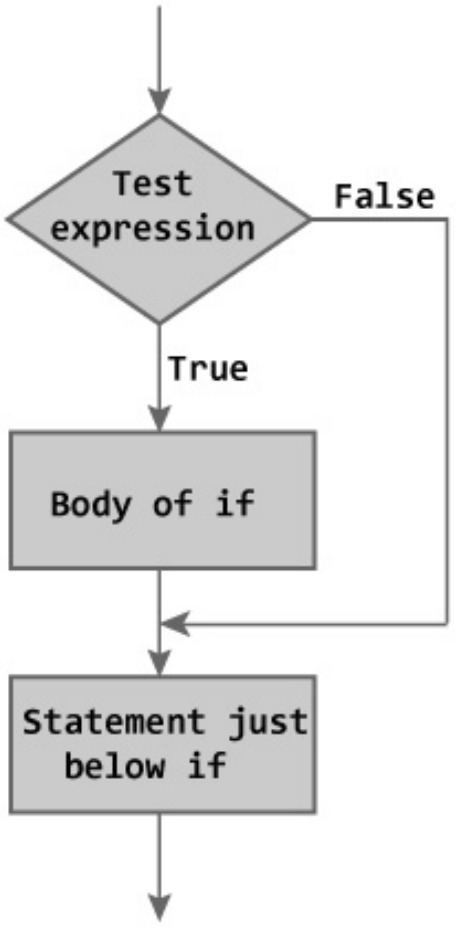
```
/* Program to display a number
   if user enters negative number.
   If user enters positive number,
   that number won't be displayed. */

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

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (number < 0)
    {
        printf("You entered %d.\n", number);
    }

    printf("The if statement is easy.");
}
```

Single Selection : `if` construct

Example :

```
/* Program to display a number
   if user enters negative number.
   If user enters positive number,
   that number won't be displayed. */

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

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (number < 0)
    {
        printf("You entered %d.\n", number);
    }

    printf("The if statement is easy.");
}
```

Semantics (meaning) :



Single Selection : `if` construct

Example :

```
/* Program to display a number
   if user enters negative number.
   If user enters positive number,
   that number won't be displayed. */

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

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (number < 0)
    {
        printf("You entered %d.\n", number);
    }

    printf("The if statement is easy.");
}
```

Single Selection : `if` construct

Example :

```
/* Program to display a number
   if user enters negative number.
   If user enters positive number,
   that number won't be displayed. */

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

    printf("Enter an integer: ");
    scanf("%d", &number);

    if (number < 0)
    {
        printf("You entered %d.\n", number);
    }

    printf("The if statement is easy.");
}
```

Output :

```
Enter an integer: -2
You entered -2.
The if statement is easy.

-------------------------

Enter an integer: 5
The if statement in C programming is easy.
```

Double Selection : `if-else` construct

Decide to execute a part of the program based on a condition is **true** and some other part if condition is **false**.

Double Selection : `if-else` construct

Decide to execute a part of the program based on a condition is **true** and some other part if condition is **false**.

Eg : If $b^2 - 4ac$ negative, we should report that the quadratic has no real roots.

Double Selection : `if-else` construct

Decide to execute a part of the program based on a condition is **true** and some other part if condition is **false**.

Eg : If $b^2 - 4ac$ negative, we should report that the quadratic has no real roots.
Syntax :

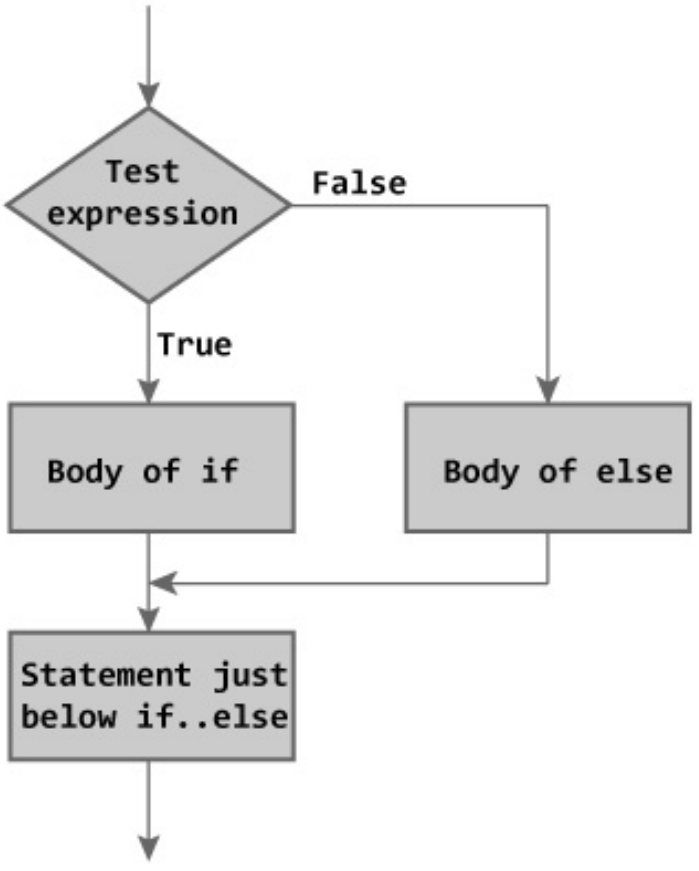
```
if (condition)
{ Statement Sequence 1 }
else
{ Statement Sequence 2 }
```

Double Selection : `if-else` construct

Decide to execute a part of the program based on a condition is **true** and some other part if condition is **false**.

Eg : If $b^2 - 4ac$ negative, we should report that the quadratic has no real roots.

Semantics (meaning) :



Syntax :

```
if (condition)
{ Statement Sequence 1 }
else
{ Statement Sequence 2 }
```

Double Selection : `if-else` construct - Example

Example :

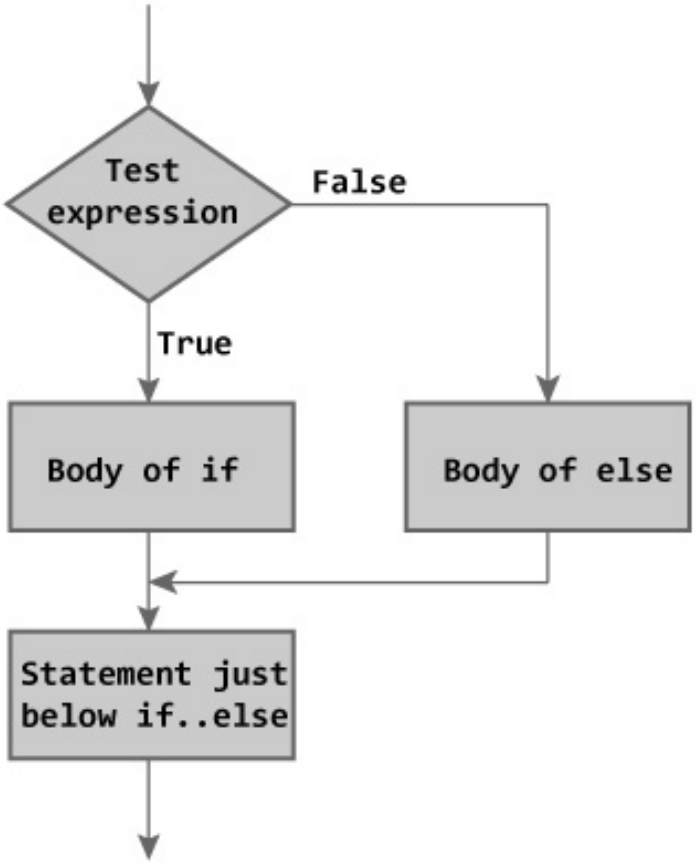
```
// Program to check whether an
// integer entered by the user
// is odd or even

#include <stdio.h>
int main()
{
    int number;
    printf("Enter an integer: ");
    scanf("%d",&number);

    // True if remainder is 0
    if( number%2 == 0 )
        printf("%d is an even integer.",number);
    else
        printf("%d is an odd integer.",number);
    return 0;
}
```

Double Selection : `if-else` construct - Example

Example :

```
// Program to check whether an
// integer entered by the user
// is odd or even

#include <stdio.h>
int main()
{
    int number;
    printf("Enter an integer: ");
    scanf("%d",&number);

    // True if remainder is 0
    if( number%2 == 0 )
        printf("%d is an even integer.",numbe
    else
        printf("%d is an odd integer.",numbe
    return 0;
}
```

Semantics (meaning) :



How do we specify conditions?

- Specified using relational and equality operators.

How do we specify conditions?

- Specified using relational and equality operators.
- Relational: $>, <, >=, <=$

How do we specify conditions?

- Specified using relational and equality operators.
- Relational: $>, <, >=, <=$
- Equality: $==, !=$

How do we specify conditions?

- Specified using relational and equality operators.
- Relational: $>, <, >=, <=$
- Equality: $==, !=$
- Usage: for a,b values or variables
  $a > b, a < b, a >= b, a <= b, a == b, a\ != b$

How do we specify conditions?

- Specified using relational and equality operators.
- Relational: $>, <, >=, <=$
- Equality: $==, !=$
- Usage: for a,b values or variables
  $a > b, a < b, a >= b, a <= b, a == b, a\ != b$
- A condition is satisfied or true, if the relational operator, or equality is satisfied.

How do we specify conditions?

- Specified using relational and equality operators.
- Relational: $>, <, >=, <=$
- Equality: $==, !=$
- Usage: for a,b values or variables
  $a > b, a < b, a >= b, a <= b, a == b, a\ != b$
- A condition is satisfied or true, if the relational operator, or equality is satisfied.
- For a = 3, and b = 5:
  - $a < b, a <= b$, and $a\ != b$ are **true**.
  - $a > b, a >= b, a == b$ are **false**.

How do we specify conditions?

- Specified using relational and equality operators.
- Relational: $>, <, >=, <=$
- Equality: $==, !=$
- Usage: for a,b values or variables
  $a > b, a < b, a >= b, a <= b, a == b, a\ != b$
- A condition is satisfied or true, if the relational operator, or equality is satisfied.
- For a = 3, and b = 5:
  - $a < b, a <= b$, and $a\ != b$ are **true**.
  - $a > b, a >= b, a == b$ are **false**.
- Expression can contain relational, logical or equality operators.

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

An Example Problem

Accept a character from {W, A, B} and output appropriate message about the grade.

An Example Problem

Accept a character from {W, A, B} and output appropriate message about the grade.

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

     char input;

     printf("Input a character:\t" );
     scanf ("%c", &input);

     if (input == 'W') {
        printf("Attendance is below 85 %%\n");
     }
     if (input == 'A') {
        printf("Marks between 90--100 %%\n");
     }
     if (input == 'B') {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;
}
```

An Example Problem

Accept a character from {W, A, B} and output appropriate message about the grade.

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

     char input;

     printf("Input a character:\t" );
     scanf ("%c", &input);

     if (input == 'W') {
        printf("Attendance is below 85 %%\n");
     }
     if (input == 'A') {
        printf("Marks between 90--100 %%\n");
     }
     if (input == 'B') {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;
}
```

Program prints error message even when we enter valid characters 'W' or 'A'.

An Example Problem

Accept a character from {W, A, B} and output appropriate message about the grade.

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

     char input;

     printf("Input a character:\t" );
     scanf ("%c", &input);

     if (input == 'W') {
        printf("Attendance is below 85 %%\n");
     }
     if (input == 'A') {
        printf("Marks between 90--100 %%\n");
     }
     if (input == 'B') {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;
}
```

Program prints error message even when we enter valid characters 'W' or 'A'.

A correct program.

Accept a character from {W, A, B} and output appropriate message about the grade.

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

     char input;

     printf("Input a character :\t" );
     scanf ("%c", &input);

     if (input == 'W') {
        printf("Attendance is below 85 %%\n");
     }
     else if (input == 'A') {
        printf("Marks between 90--100 %%\n");
     }
     else if (input == 'B') {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;

}
```

Is this correct?

Accept a character from {W, A, B} and output appropriate message.

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

     char input, W, A, B;

     printf("Input a character :\t" );
     scanf ("%c", &input);

     if (input == W) {
        printf("Attendance is below 85 %%\n");
     }
     else if (input == A) {
        printf("Marks between 90--100 %%\n");
     }
     else if (input == B) {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;
}
```

Is this correct?

Accept a character from {W, A, B} and output appropriate message.

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

     char input, W, A, B;

     printf("Input a character :\t" );
     scanf ("%c", &input);

     if (input == W) {
        printf("Attendance is below 85 %%\n");
     }
     else if (input == A) {
        printf("Marks between 90--100 %%\n");
     }
     else if (input == B) {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;
}
```

Notice the variables W, A, B declared. What is the output of the program?

Is this correct?

Accept a character from {W, A, B} and output appropriate message.

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

     char input, W, A, B;

     printf("Input a character :\t" );
     scanf ("%c", &input);

     if (input == W) {
        printf("Attendance is below 85 %%\n");
     }
     else if (input == A) {
        printf("Marks between 90--100 %%\n");
     }
     else if (input == B) {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;
}
```

Notice the variables W, A, B declared. What is the output of the program?

variable vs character constant

- if (input == W)
  - comparing a variable input with another variable W.
  - What is the value of the variable W?

variable vs character constant

- if (input == W)
  - comparing a variable input with another variable W.
  - What is the value of the variable W?
  - If W is a character and is initialized to W, you will have desired behaviour.

variable vs character constant

- if (input == W)
  - comparing a variable input with another variable W.
  - What is the value of the variable W?
  - If W is a character and is initialized to W, you will have desired behaviour.
- if (input == 'W')
  - comparing a variable input with character constant W.

variable vs character constant

- if (input == W)
  - comparing a variable input with another variable W.
  - What is the value of the variable W?
  - If W is a character and is initialized to W, you will have desired behaviour.
- if (input == 'W')
  - comparing a variable input with character constant W.

- In C, we can define some variables to be constants as well.
  - const float PI = 3.14;
  - const int myConstant = 71289;
  - const char gradeW = 'W';
- Recall what are valid variables names.

Are the parenthesis needed?

Accept a character from {W, A, B} and output appropriate message.

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

     char input;

     printf("Input a character :\t" );
     scanf ("%c", &input);

     if (input == 'W') {
        printf("Attendance is below 85 %%\n");
     }
     else if (input == 'A') {
        printf("Marks between 90--100 %%\n");
     }
     else if (input == 'B') {
        printf("Marks between 80--90 %% \n");
     }
     else {
        printf("Invalid Character. Enter one of W, A, B\n");
     }
     return 0;

}
```

How is the nesting?

Accept a character from {W, A, B} and output appropriate message.

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

  char input;

  printf("Input a character :\t" );
  scanf ("%c", &input);

  if (input == 'W')
    printf("Attendance is below 85 %%\n");

  else {
    if (input == 'A')
      printf("Marks between 90--100 %%\n");
    else {
      if (input == 'B')
printf("Marks between 80--90 %% \n");
      else
printf("Invalid Character. Enter one of W, A, B\n");
    }
  }
  return 0;

}
```

---

if else: example2

If a student gets less than 40 marks, report that s/he has to repeat the course. If student gets greater than 75 marks, report that s/he got distinction.

if else: example2

If a student gets less than 40 marks, report that s/he has to repeat the course. If student gets greater than 75 marks, report that s/he got distinction.

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

     printf("Enter your marks: \t");
     scanf("%d", &marks);

     if (marks > 40)
        if (marks > 75)
           printf("You got distinction\n");
     else
        printf("You need to repeat the course\n");
}
```

if else: example2

If a student gets less than 40 marks, report that s/he has to repeat the course. If student gets greater than 75 marks, report that s/he got distinction.

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

     printf("Enter your marks: \t");
     scanf("%d", &marks);

     if (marks > 40)
        if (marks > 75)
           printf("You got distinction\n");
     else
        printf("You need to repeat the course\n");
}
```

- No errors during compilation or execution.
- Does not produce desired behaviour.
- else pairs with the latest unpaired if.
- referred to as a "dangling else problem."

if else: example2 – correct program

If a student gets less than 40 marks, report that he has to repeat the course. If student gets greater than 75 marks, report that he got distinction.

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

     printf("Enter your marks: \t");
     scanf("%d", &marks);

     if (marks > 40) {
        if (marks > 75)
           printf("You got distinction\n");
     }
     else
        printf("You need to repeat the course\n");
}
```

- Draw the control flow of the program.

if else: example2 – observe carefully

If a student gets less than 40 marks, report that he has to repeat the course. If student gets greater than 75 marks, report that he got distinction.

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

     printf("Enter your marks: \t");
     scanf("%d", &marks);

     if (marks > 40) {
        if (marks > 75);
           printf("You got distinction\n");
     }
     else
        printf("You need to repeat the course\n");
}
```

if else: example2 – observe carefully

If a student gets less than 40 marks, report that he has to repeat the course. If student gets greater than 75 marks, report that he got distinction.

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

     printf("Enter your marks: \t");
     scanf("%d", &marks);

     if (marks > 40) {
        if (marks > 75);
           printf("You got distinction\n");
     }
     else
        printf("You need to repeat the course\n");
}
```

- What is the output of the program on
  - 40, 50, 75, 85

if else: example2 – observe carefully

If a student gets less than 40 marks, report that he has to repeat the course. If student gets greater than 75 marks, report that he got distinction.

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

     printf("Enter your marks: \t");
     scanf("%d", &marks);

     if (marks > 40) {
        if (marks > 75);
           printf("You got distinction\n");
     }
     else
        printf("You need to repeat the course\n");
}
```

- What is the output of the program on
  - 40, 50, 75, 85
- Note the semicolon if (marks $>$ 75);
- Semicolon is a statement terminator.