Control Statements

Lecture 5

Control Statements

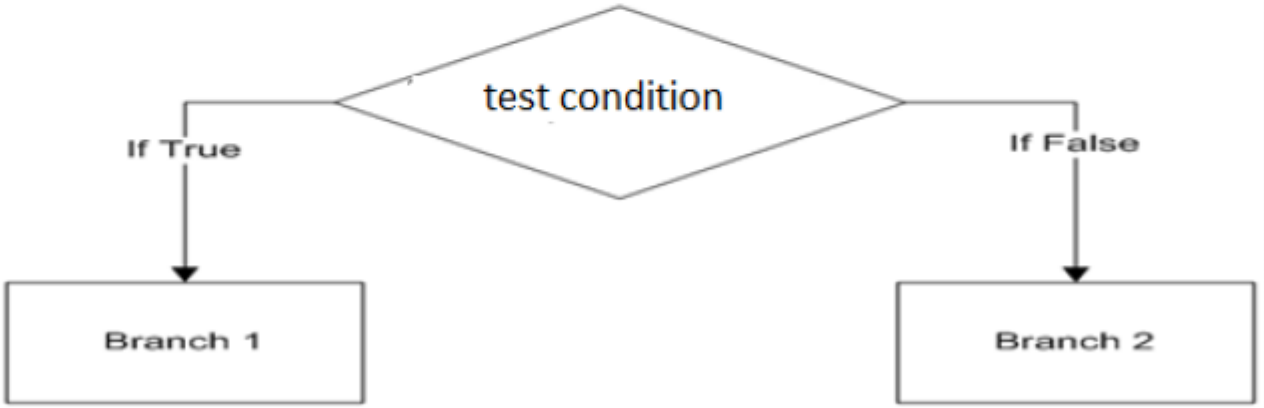
- Needed while there is to perform repeated actions or skips some statements.
- Alter the flow of execution of the programs.
- Two types:

  1) Decision making statements

  2) Loop constructs

Decision Making Statements

- Programs should be able to make logical (true/false) decisions based on given condition and decision making statement is used for these programs
- Also called branching or conditional statement



- Decision making statement
    1. if statement
    2. if-else statement
    3. switch statement

If Statement

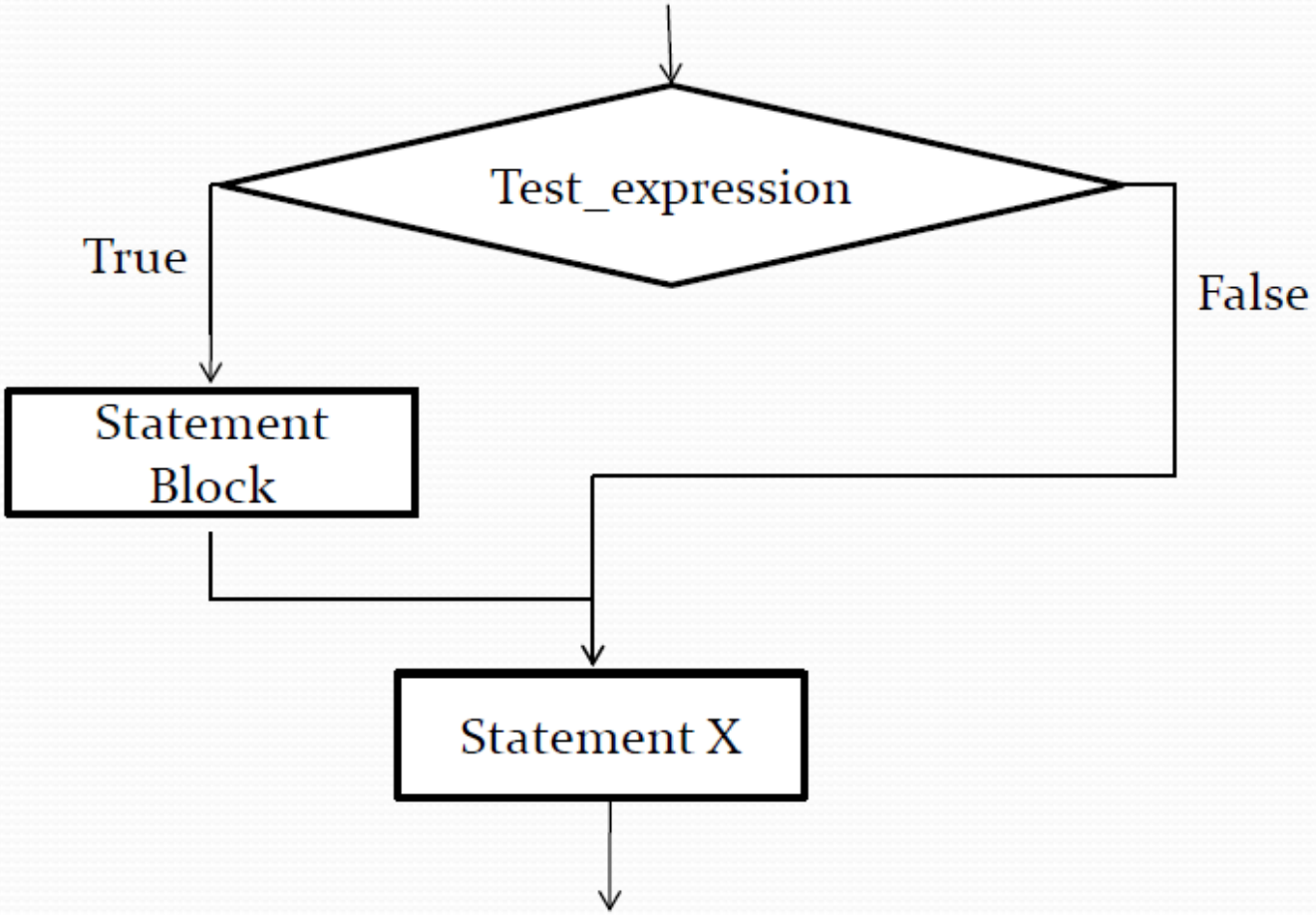
- Syntax

```
if(test_expression )
{
        statement A;
        statement B;
        ....
}
 statement X;
```

- If test_expression is true statement block will be executed, otherwise execution will jump to statement X

If Statement

- Flowchart



If Statement

- Example:

```
if(a>5)
{
        printf("a is greater than 5");
}
printf("a=%d",a);
```

If else statement

- Extension of if statement
- Syntax

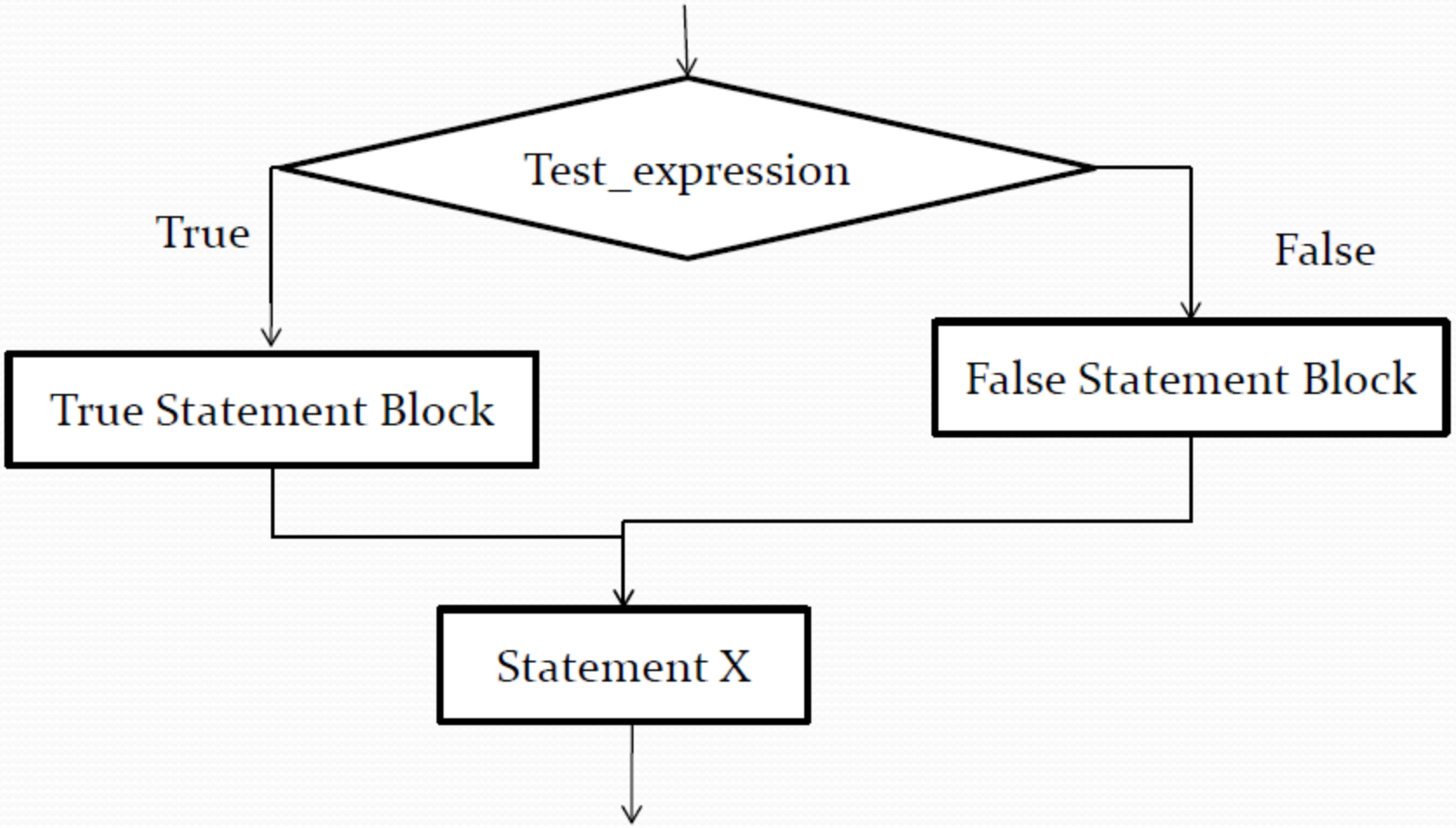
```
if(test_expression )
{
        statement A;
        statement B;
        ....
}
else
{
        statement M;
        statement N;
        ....
}
 statement X;
```

True Statement block

False Statement block

If else Statement

- Flowchart



If else Statement

- Example:

```
if(a>5)
{
        printf("a is greater than 5");
}
else
{
        printf("a is less than or equal to 5");
}
printf("a=%d",a);
```

Nested if-else Statement

```
if(a>5)
{
        if(a>10)
                printf("a is greater than 10");
        else
                printf("a is greater than 5");
}
else
{
        if(a==5)
                printf("a is equal to 5");
        else
                printf("a is less than 5");
}
printf("a=%d",a);
```

Outer if-statement block

Outer else-statement block

Else-if ladder

- It is another way of putting ifs together when multipath decisions are involved

```
if(a==5)
{
        printf("a is equal to 5");

}
else if(a>5)
{
        printf("a>5");
}
else
{
        printf("a < 5");
}
printf("a=%d",a);
```

```
if(a==5)
        printf("a is equal to 5");
else
{
        if(a<5)
                printf("a > 5");
        else
                printf("a <5");
}
printf("a=%d",a);
```

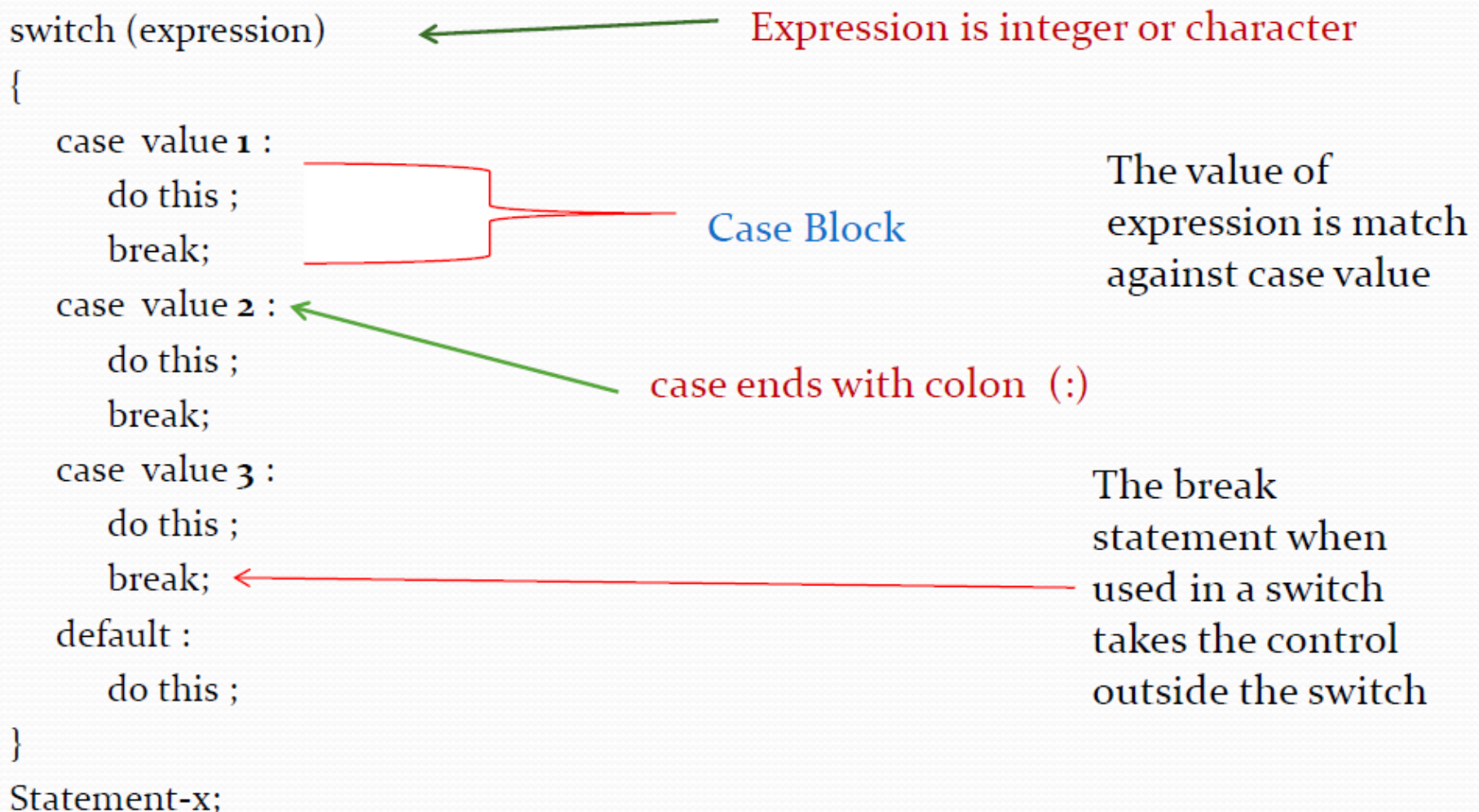
Else-if ladder

- Conditions are executed from top to bottom
- As soon as true statement is found statement associated with it is executed and control is transferred to first statement after else-if, skipping rest of ladder
- When all statement are false final else statement is executed

- Write a program to read percentage of a student and print the equivalent grade
  - Between 100 to 90 : grade A
  - Between 89 to 80 : grade A-
  - Between 79 to 70 : grade B-
  - Between 69 to 60 : grade B-
  - Between 59 to 50 : grade c
  - Below 50 : Fail

Switch Statement

- The control statement that allows us to make a decision from the number of choices is called a **switch**, or more correctly a **switch-case-default**
- Syntax



Switch Statement

```
char single;
printf("Enter a character");
scanf("%c",& single);
switch(single)
{
        case 'x' :
                printf("x");
                break;
        case 'y' :
                printf("y");
                break;
        case 'z' :
                printf("z");
                break;
        default:
                printf("Not  x and y and z \n");

}
printf("done");
```

Switch Statement

- Above example is equivalent to following if-else-if statement

```
char single;
printf("Enter a character");
scanf("%c",& single);

if(single==‘x’)
        printf(“x");
else if(single==‘y’)
        printf(“y");
else if(single==‘z’)
        printf(“z");
else
        printf(“Not x and y and z \n");

printf(“done”);
```

Switch Statement

- If we have no default case, then the program simply falls through the entire switch and continues with the next instruction (if any,) that follows the closing brace of switch.
- float expression cannot be tested using a **switch**