

The **if** statement

Relational operators :

Per>=75 distinction

General syntax of if statement :

if(condition)

{ body of if statment

Action.....

}

```
if (testScore > 60)
    printf( "You pass");
```

```
if (testScore > 99)
{
    printf( "You pass");
    printf("you will be honored by the college");
    printf(" you will be awarded with gold medal");
}
```

if the if condition is true and it has to execute only one statement then there should not be need of curly brackets to enclose the body

but, if the if condition is true and it has to execute more than one statement then there should be need of curly brackets to enclose the body

The **if/else** statement

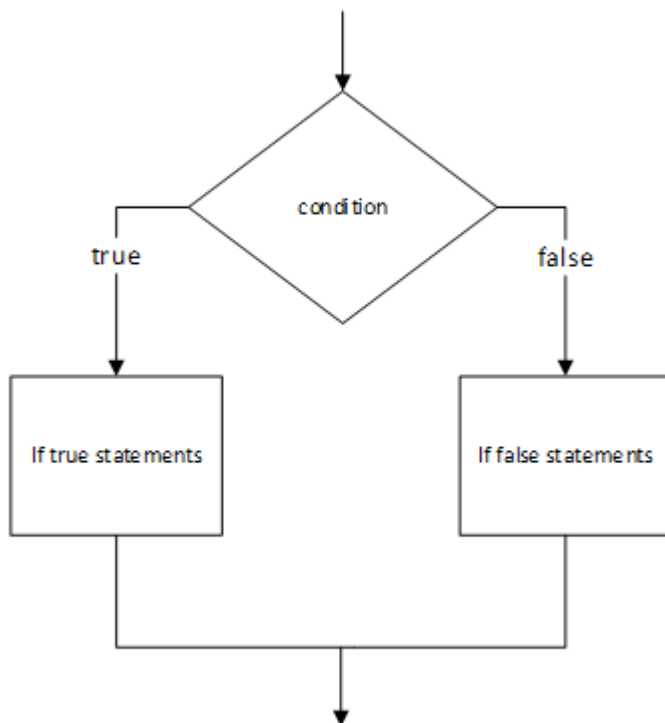
The if/else statement extends the if statement by specifying an action if the if (*true/false expression*) is false.

```
if (condition)
{
    // do this if condition is true
```

```
// if true statements
}  
else  
{  
    // do this is condition is false  
    // if false statements  
}
```

With the if statement, a program will execute the true code block or do nothing. With the if/else statement, the program will execute either the true code block or the false code block so something is always executed with an if/else statement.

Flow chart view of if/else



Where to use two statements versus one if/else statement

Use two if statements if both if statement conditions could be true at the same time. In this example, both conditions can be true. You can pass and do great at the same time.

Use an if/else statement if the two conditions are mutually exclusive meaning if one condition is true the other condition must be false.

```
if (testScore > 60)  
    cout << "You pass" << endl;
```

```
if (testScore > 90)
    cout << "You did great" << endl;
```

For example, before noon (AM) and after noon (PM) are mutually exclusive. It is either one or the other. Using a 24 hour time system (12 is noon and 24 is midnight), if the time of day is ≥ 12 it is PM, else it is AM.

```
if (timeOfDay >=12)
    cout << "PM" << endl;

else // it must be AM
    cout << "AM" << endl;
```

Curly brackets with if/else statements

The else part of the if/else statement follows the same rules as the if part. If you want to execute multiple statements for the else condition, enclose the code in curly brackets. If you only need to execute a single statement for the else condition, you do not need to use curly brackets.

```
if (condition)
{
    multiple
    statements
}
else
    single statement
```

```
if (condition)
{
    multiple
    statements
}
else
{
    multiple
    statements
}
```

```
}
```

Eg.

- 1) Program to check whether entered no is positive or negative using if statements.

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()
```

```
{
```

```
int n ;
```

```
printf("enter the value of n\n");
```

```
scanf("%d", &n);
```

```
if(n>0)
```

```
printf("\nthe number is positive");
```

```
if(n<0)
```

```
printf("\nthe number is negative");
```

```
getch();
```

```
}
```

2) Program to check whether entered no is positive or negative using if statements.

```
#include<stdio.h>

#include<conio.h>

void main()

{

int n ;

printf("enter the value of n\n");

scanf("%d", &n);

if(n>0)

printf("\nthe number is positive");

if(n<0)

printf("\nthe number is negative");

if(n==0)

printf("\nthe number is zero");

getch();

}
```

3) Program to check whether entered no is positive or negative using if-else statements.

```
#include<stdio.h>

#include<conio.h>

void main()

{

int n ;

printf("enter the value of n\n");

scanf("%d", &n);
```

```
if(n>=0)
printf("\nthe number is positive");
else
printf("\nthe number is negative");
getch();
}
```

3) Program to check whether entered no is even or odd using if-else statements.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n ;
printf("enter the value of n\n");
scanf("%d", &n);
if(n%2==0)
printf("\nthe number is even");
else
printf("\nthe number is odd");
getch();
}
```

The **if/else if** statement

The if/else if statement allows you to create a chain of if statements. The if statements are evaluated in order until one of the if expressions is true or the end of

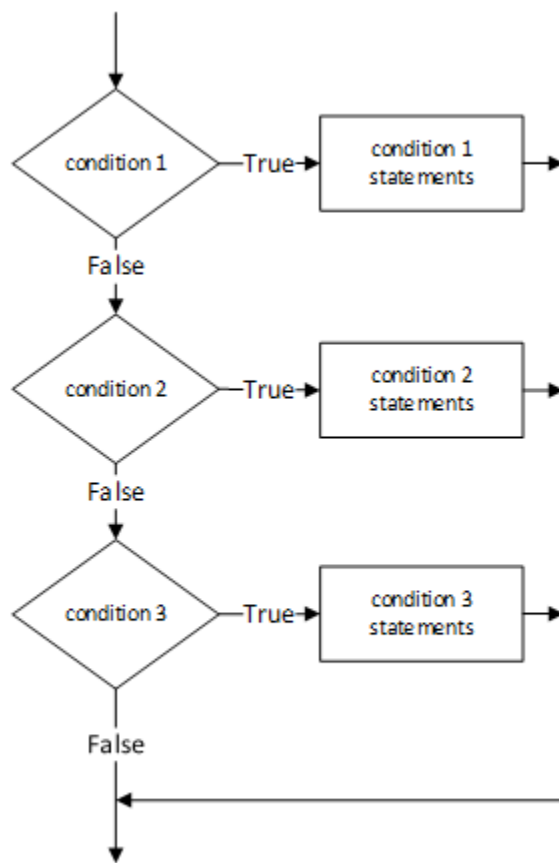
the if/else if chain is reached. If the end of the if/else if chain is reached without a true expression, no code blocks are executed.

```
if (condition1)
{
    // do this if condition1 is true
    // condition 1 statements
    // then exit if/else if
}
else
    if (condition2)
    {
        // do this if condition2 is true
        // condition 2 statements
        // then exit if/else if
    }
    else
        if (condition3)
        {
            // do this if condition3 is true
            // condition3 statements
            // then exit if/else if
        }

        Else
        {
        }

// continuation point after if/else if is complete
```

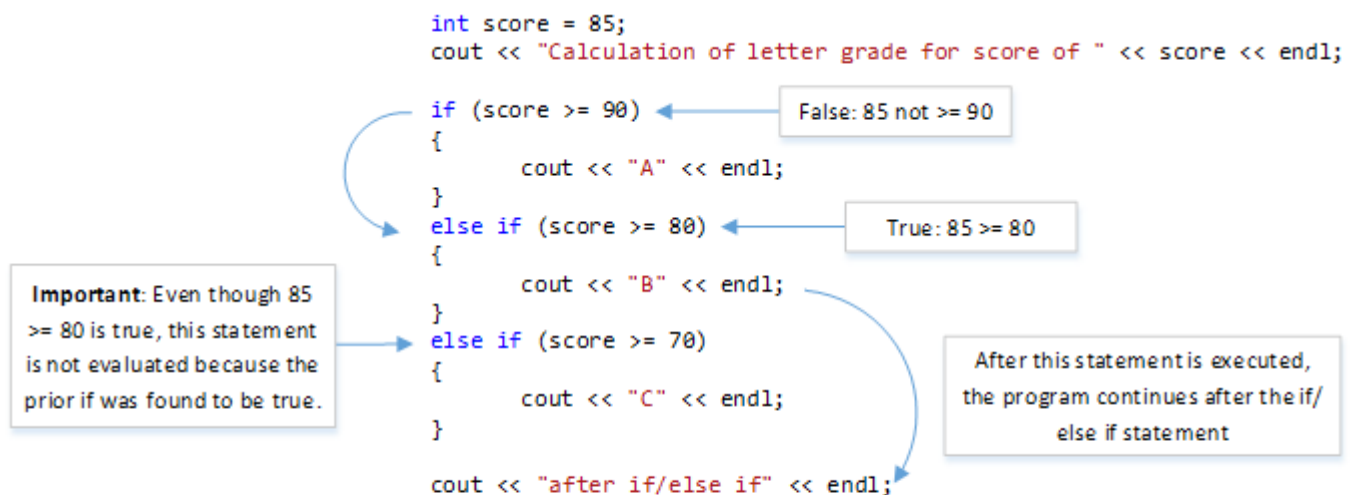
Flow chart view



if/else if flow control

Note: It is very important to understand that once a condition is found to be true, **no other if statements are evaluated** and once the code block for the true statement is completed, the program continues from the end of the if/else if statement.

Let's look at an example of this.



```

Calculation of letter grade for score of 85
B
after if/else if
  
```

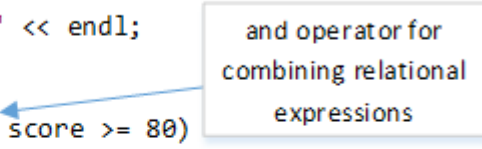

Where to use if/else if

Use the “if/else if” if you need to choose either one or none of a series of options. You could write the above program using separate if statements such as show below but the if/else if approach is cleaner and better indicates that the program should choose either one or none of the options.

```
if (score >= 90)
{
    cout << "A" << endl;
}

if (score < 90 && score >= 80)
{
    cout << "B" << endl;
}

if (score < 80 && score >= 70)
{
    cout << "C" << endl;
}
```

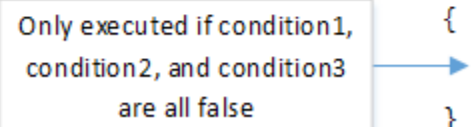


and operator for combining relational expressions

Add a trailing else to the if/else if statement

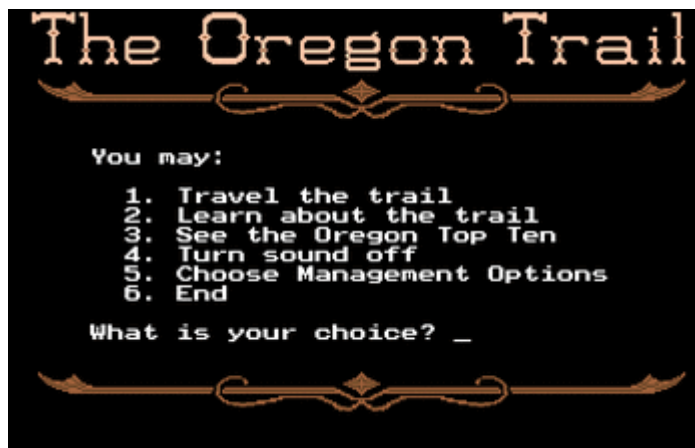
You can add a trailing else statement to the if/else if statement if you want to execute code if none of the if statements are true.

```
if (condition1)
{
    // do this if condition1 is true
    // condition1 statements
    // then exit if/else if
}
else if (condition2)
{
    // do this if condition2 is true
    // condition2 statements
    // then exit if/else if
}
else if (condition3)
{
    // do this if condition3 is true
    // condition3 statements
    // then exit if/else if
}
else // if no if condition was true
{
    // do this if no if conditions were true
    // trailing else statements
}
```



Only executed if condition1, condition2, and condition3 are all false

Menu driven programs



The C++ *Early Objects* book contains a discussion (p. 181) of menu driven programs. In a menu driven program, the user is presented with a menu or series of options. The user then selects an option and the program proceeds with execution based on the menu selection. if/else if statements are often used to control program flow in menu driven programs.

Nested if statements

A nested if statement is an if statement placed inside another if statement. Nested if statements are often used when you must test a combination of conditions before deciding on the proper action.

```

    if (isRaining)
    {
        if (temp > 45)
            cout << "Wear light weight rain coat" << endl;
        else
            cout << "Wear fleece and rain coat " << endl;
    }
    else if (isSnowing)
    {
        if (temp > 20)
        {
            cout << "Wear soft shell jacket" << endl;
        }
        else if (temp > 0)
        {
            cout << "Wear down jacket" << endl;
        }
        else
        {
            cout << "Wear base layers and down jacket" << endl;
        }
    }
    else
        cout << "It is hard to come up with interesting examples" << endl;

```

Outer if/
else if

Inner if/
else if

How do you match else's with if's

An else statement is matched to the *closest previous if statement* that does not already have its own else statement.

Proper indenting makes it easier to see the matching. Just remember that in an if or else statement, if the expression is true, the program executes the following curly bracketed code block or single statement.

Eg. 1) Write a program to find the maximum number among three numbers using if else if block.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
int a,b,c ;
```

```
printf("enter the value of a,b,c\n");
```

```
scanf("%d%d%d", &a,&b,&c);
```

```
if(a>b && a>c)
```

```
printf("\nthe number %d is max",a);
```

```
else if(b>c)
```

```
printf("\nthe number %d is max",b);
```

```
else
```

```
printf("\nthe number %d is max",c);
```

```
return 0;
```

```
}
```