COMPUTER PROGRAMMING - I

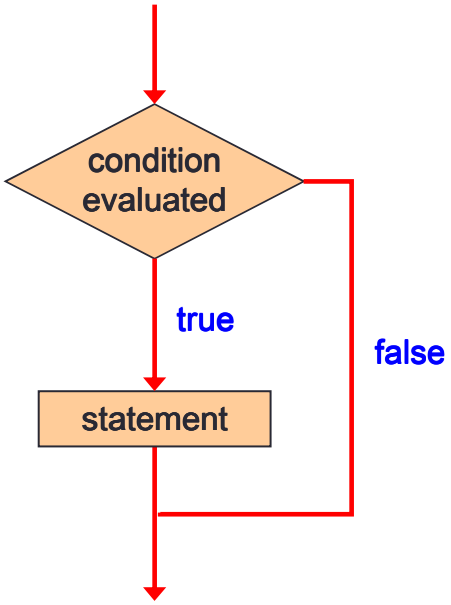
Flow of Control

- The order in which statements are executed is called **program control/flow of control**.
- Unless specified otherwise, the order of statement execution through a function is linear: one statement after another in sequence
- There are three types of program controls:
- **Sequence** control structure.
- **Selection structures** such as if, if-else, nested if, if-if-else, if-else-if and switch-case-break.
- **Repetition** (loop) such as for, while and do-while

Conditional Statements

- A *conditional statement* lets us choose which statement will be executed next. Therefore they are sometimes called *selection statements*
- The C conditional statements are the:
  - *if statement*
  - *if-else statement*
  - *switch statement*

Logic of an if statement



The if Statement

- The *if statement* has the following syntax:

The *condition* must be a boolean expression. It must evaluate to either true or false.

`if` is a C reserved word

```
if ( condition )
    statement;
```

If the *condition* is true, the *statement* is executed. If it is false, the *statement* is skipped.

Program using if

```
int main ()
{
int a = 100;

if( a % 2 ==0 )
  {
        printf("a is a even number\n" );
  }
    printf("value of a is : %d\n", a);

}
```

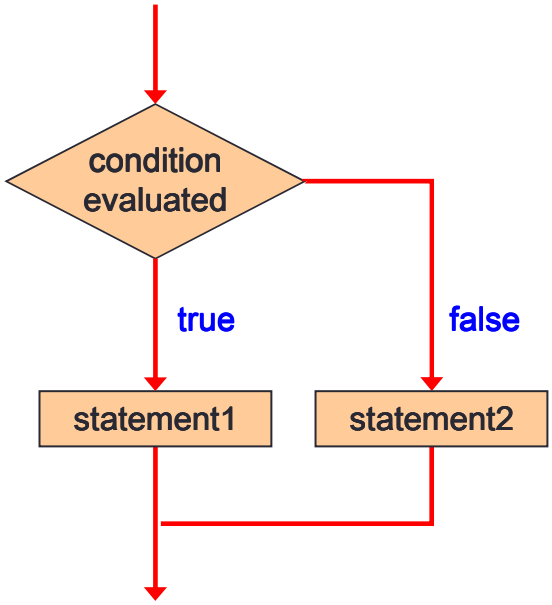
Practice time

- WAP to input two numbers and print equal if both numbers are equal

```
int main ()
{
    int num1,num2;
    printf("Enter two numbers");
    scanf("%d %d",&num1,&num2);
   if( num1 == num2 )
   {
          printf("Numbers are equal");
   }


}
```

Logic of an if-else statement



The if-else Statement

- An *else clause* can be added to an `if` statement to make an *if-else statement*

```
if ( condition )
    statement1;
else
    statement2;
```

- If the condition is true, statement1 is executed; if the condition is false, statement2 is executed
- One or the other will be executed, but not both

Program using if-else

```
void main ()
{
int a = 100;
 if( a < 20 )
   {
          printf("a is less than 20\n" );
   }
   else
   {
         printf("a is not less than 20\n" );
   }
   printf("value of a is : %d\n", a);
 }
```

Boolean Expressions

- A condition often uses one of C's *equality operators* or *relational operators*, which all return boolean results:

==	equal to
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

- Note the difference between the equality operator (==) and the assignment operator (=)

Boolean Expressions in C

- C does not have a boolean data type.
- Therefore, C compares the values of variables and expressions against 0 (zero) to determine if they are true or false.
- If the value is 0 then the result is implicitly assumed to be false.
- If the value is different from 0 then the result is implicitly assumed to be true.

Quiz time

- 1. if(3+2%5)
- printf(“This works”);
- 2. if (a=10)
- printf(“Even this works”);
- 3. if(3.15)
- printf(“Even this also works”);
- 4.if(-5)
- printf(“ Surprisingly even this also works”);

Quiz time

- What will be output of the program
- int main()
- {
- int i;
- printf(“Enter value of I”);
- scanf(“%d”,&i);
- if (i=5)
- printf(“You entered 5”);
- else
- printf(“You entered something other than 5”);
- }
- Suppose i = 200

Quiz time

- What will be output of the following program
- int main()
- {
- int a =300,b,c;
- if(a>=400)
- b=300;
- c= 200;
- printf(“\n%d %d”,b,c);
- }

Quiz time

- What will be output of the following program
- int main()
- {
- int a =500,b,c;
- if(a>=400)
- b=300;
- c= 200;
- printf(“\n%d %d”,b,c);
- }

Quiz time

- What will be output of the following program
- int main()
- {
- int x =3, y=5;
- if(x==3)
- printf(“\n%d ”,x);
- else;
- printf(“\n%d ”,y);
- }

Quiz time

- What will be output of the following program
- int main()
- {
- int x =3;
- float y=3.0;
- if(x==y)
- printf(“\nx and y are equal”);
- else
- printf(“\nx and y are not equal”);
- }

Block Statements

- Several statements can be grouped together into a *block statement* delimited by braces

```
if (total > MAX)
{
   printf ("Error!!\n");
   errorCount++;
}
```

Block Statements

- In an `if-else` statement, the `if` portion, or the `else` portion, or both, could be block statements

```
if (total > MAX)
{
   printf("Error!!");
   errorCount++;
}
else
{
   printf ("Total: %d", total);
   current = total*2;
}
```

Practice time

- In a company an employee is paid as under:
- If his basic salary is less than Rs.1500, then HRA =10% of basic salary and DA = 90% of basic salary.
- If his salary is either equal to or above Rs.1500 the HRA = Rs.500 and DA = 98% of basic salary.
- If the employee basic salary is input through keyboard write a program to find his gross salary(gross salary = basic salary +hra + da)

Practice time

```
main( )
{
float bs, gs, da, hra ;
printf ( "Enter basic salary " ) ;
scanf ( "%f", &bs ) ;
if ( bs < 1500 )
{
hra = bs * 10 / 100 ;
da = bs * 90 / 100 ;
}
```

Practice time

- else
- {
- hra = 500 ;
- da = bs * 98 / 100 ;
- }
- gs = bs + hra + da ;
- printf ("gross salary = Rs. %f", gs) ;
- }

Nested if Statements

- The statement executed as a result of an `if` statement or `else` clause could be another `if` statement
- These are called *nested if statements*
- An *else* clause is matched to the last unmatched `if` (no matter what the indentation implies)
- Braces can be used to specify the `if` statement to which an *else* clause belongs

```
#include <stdio.h>

int main ()
{
int a = 100;

if( a == 10 )
   {
         printf("Value of a is 10\n" );
   }
   else if( a == 20 )
   {
           printf("Value of a is 20\n" );
   }
```

```
else if( a == 30 )
   {
       printf("Value of a is 30\n" );
   }
   else
   {
           printf("None of the values is matching\n" );
   }
   printf("Exact value of a is: %d\n", a );

   return 0;
}
```

Program using nested if

```
int main()
{
int age;
printf( "Please enter your age" );
scanf( "%d", &age );
if ( age < 100 )
{
            printf ("You are pretty young!\n" );
}
 else if ( age == 100 )
{
          printf( "You are old\n" ); }
```

Program using nested if

```
else
{
printf( "You are really old\n" );
}
return 0; }
```

Practice time

- **WAP to find whether a number is +ve,-ve or 0**

```
int main()
{

    int num;
    printf("Enter any number");
    scanf("%d",&num);
    if(num>0)
      printf("Number is positive");
    else if(num<0)
      printf("Number is negative");
    else
      printf("Number is 0");
}
```

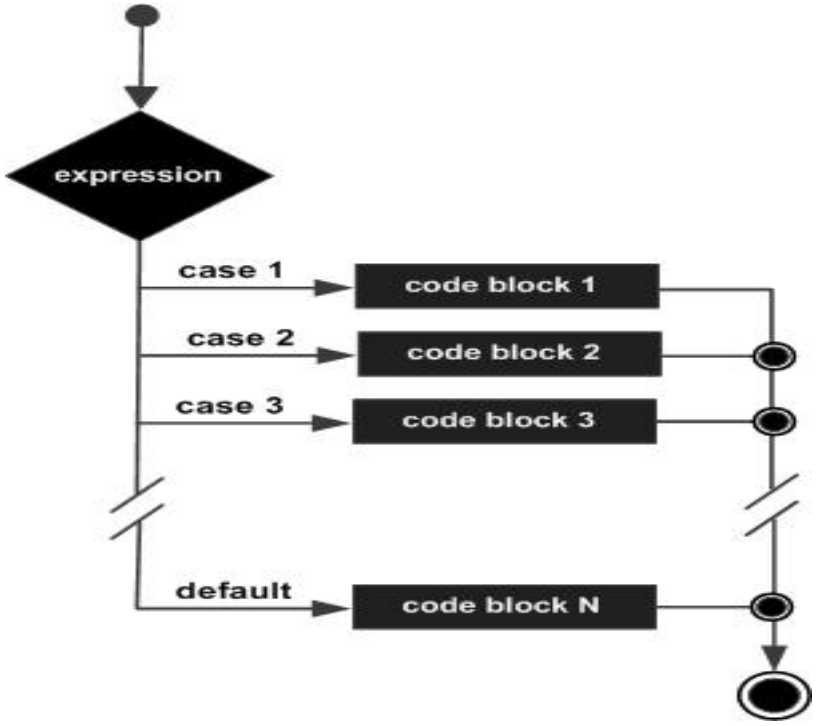
The switch Statement

- The *switch statement* provides another way to decide which statement to execute next
- The *switch* statement evaluates an expression, then attempts to match the result to one of several possible *cases*
- Each case contains a value and a list of statements
- The flow of control transfers to statement associated with the first case value that matches

Switch

- switch (integer expression)
- {
- case constant 1 :
- do this ;
- case constant 2 :
- do this ;
- case constant 3 :
- do this ;
- default :
- do this ;
- }

- First, the integer expression following the keyword **switch** is evaluated.
- The value it gives is then matched, one by one, against the constant values that follow the **case** statements.
- When a match is found, the program executes the statements following that **case**, and all subsequent **case** and **default** statements as well.
- If no match is found with any of the **case** statements, only the statements following the **default** are executed.




```
main( )
{
int i = 2 ;
switch ( i )
{
case 1 :
printf ( "I am in case 1 \n" ) ;
case 2 :
printf ( "I am in case 2 \n" ) ;
case 3 :
printf ( "I am in case 3 \n" ) ;
default :
printf ( "I am in default \n" ) ;
} }
```

Output

- I am in case 2
- I am in case 3
- I am in default

- If you want that only case 2 should get executed, it is upto you to get out of the **switch** then and there by using a **break** statement.

The switch Statement

- Often a *break statement* is used as the last statement in each case's statement list
- A *break* statement causes control to transfer to the end of the *switch* statement
- If a *break* statement is not used, the flow of control will continue into the next case
- Sometimes this may be appropriate, but often we want to execute only the statements associated with one case

```
main( )
 {
  int i = 2 ;
 switch ( i )
{
 case 1 : printf ( "I am in case 1 \n" ) ;
                break ;
case 2 : printf ( "I am in case 2 \n" ) ;
                break ;
case 3 : printf ( "I am in case 3 \n" ) ;
              break ;
 default : printf ( "I am in default \n" ) ;
 }
}
```

Important tips for switch case

- **1.You can put the cases in any order you please.**

```
main( )
{
int i = 22 ;
switch ( i )
{
case 121 :
printf ( "I am in case 121 \n" ) ;
break ;
case 7 :
printf ( "I am in case 7 \n" ) ;
break ;
```

Important tips for switch case

- case 22 :
- printf ("I am in case 22 \n") ;
- break ;
- default :
- printf ("I am in default \n") ;
- }
- }

Important tips for switch case

- **2.You are also allowed to use char values in case and switch**
- main() {
- char c = 'x' ;
- switch (c)
- {
- case 'v' :
- printf ("I am in case v \n") ;
- break ;
- case 'a' :
- printf ("I am in case a \n") ;
- break ;

Important tips for switch case

- case 'x' :
- printf ("I am in case x \n") ;
- break ;
- default :
- printf ("I am in default \n") ;
- }
- }

- **3.At times we may want to execute a common set of statements for multiple cases.**
- main( )
- {
- char ch ;
- printf ("Enter any of the alphabet a, b, or c ") ;
- scanf ("%c", &ch) ; // u can also use getchar()

Important tips for switch case

- switch (ch)
- {
- case 'a' :
- case 'A' :
- printf ("a as in ashar") ;
- break ;
- case 'b' :
- case 'B' :
- printf ("b as in brain") ;
- break ;

Important tips for switch case

- case 'c' :
- case 'C' :
- printf ("c as in cookie") ;
- break ;
- default :
- printf ("wish you knew what are alphabets") ;
- }
- }

Important tips for switch case

- 4. Even if there are multiple statements to be executed in each **case** there is no need to enclose them within a pair of braces (unlike **if**, and **else**)
- 5**. Every statement in a switch must belong to some case or the other.**
- If a statement doesn't belong to any **case** the compiler won't report an error.
- However, the statement would never get executed.
- For example, in the following program the **printf( )** never goes to work.

Important tips for switch case

```
main( )
{
int i, j ;
printf ( "Enter value of i" ) ;
scanf ( "%d", &i ) ;
switch ( i )
{
 printf ( "Hello" ) ;
 case 1 :
            j = 10 ;
            break ;
case 2 :
            j = 20 ;
           break ;
} }
```

Important tips for switch case

- 6. 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**.
- The disadvantage of **switch** is that one cannot have a case in a **switch** which looks like:
- case i <= 20 :
- All that we can have after the case is an **int** constant or a **char** constant or an expression that evaluates to one of these constants. Float is also not allowed
- The **switch** statement is very useful while writing menu driven programs.

Important tips for switch case

- There are some things cannot do with a **switch** :
- A float expression cannot be tested using a **switch**
- Multiple cases cannot use same expressions. Thus the following **switch** is illegal:

Important tips for switch case

- Multiple cases cannot use same expressions. Thus the following **switch** is illegal:
- switch (a)
- {
- case 3 :
- ...
- case 1 + 2 :
- ...
- }

Practice Time

- WAP to do the following
- Input a grade from user and print
- If A –Excellent
- B,C – Well done
- D- You passed
- F- Better try again
- Otherwise : Invalid grade

Practice Time

```
Int main( )
{
char ch ;
printf ( "Enter any of the grades A, B, C,D,F " ) ;
scanf ( "%c", &ch ) ;
switch ( ch )
{
case 'A' :
printf ( "Excellent!\n" ) ;
break ;
case 'B' :
case 'C' :
printf ( "Well done\n" ) ;
break ;
```

```
case 'D' :
printf ( "You passed\n" ) ;
break ;
case 'F' :
printf ( "Better try again\n" ) ;
break ;
default :
printf ( "Invalid grade\n" ) ;
}
}
```

Quiz time : Identify errors

- 1. if(a>b);
- printf(“ a is greater than b”);

- 2. if(n%2 == 0) then
- printf(“Even”);

- 3. switch(5)
- {
  - case 5.0: printf(“Five”);
  - Case 5: printf(“Five”);
- }