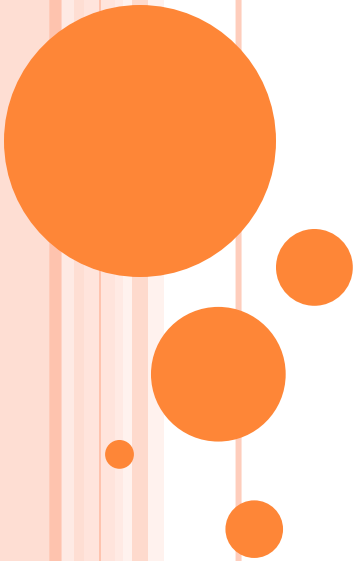


Data Type and Flow Control

Types

- Types are “categories” of values.
  - C provides four fundamental data types: char, int, float and double.
  - Other data types like arrays, pointers and structures are derived from the fundamental data types.

Character (char)

- Includes lower and upper case letters, digits, punctuation and special characters.

- Size of a character is 1 byte.

Integers (int)

- Natural nos. 0,1,2 .. and their negatives also there are octal and hexadecimal integers.
- Size of an integer is dependent on the m/c but usually 4 bytes.

Integral Types

- short
  - provides less storage, usually 2 bytes.
- long
  - provides more storage (than int but not necessary), usually 4 bytes.
- unsigned
  - have no sign.

Floating Types (float/double)

- Use to hold real values
- Three types – float (4 bytes), double (8 bytes) and long double (10 bytes).
- Suffix can be appended to specify its type – 1.2F, 1.2D, 1.2L
- Precision and range.

Data Types and their Size

Data Type	Bytes	Default Range
signed char	1	-128 to 127
Unsigned char	1	0 to 255
short signed int	2	-32768 to 32767
short unsigned int	2	0 to 65535
long signed int	4	-2147483648 to 2147483647
long unsigned int	4	0 to 4294967295
Float	4	-3.4e38 to +3.4e38
Double	8	-1.7e308 to +1.7e308
long double	10	-1.7e4932 to +1.7e4932

Type Conversion in C

- A type cast is basically a conversion from one type to another.
- **Implicit Type Conversion** Also known as '**automatic type conversion**.

  - Done by the compiler on its own, without any external trigger from the user.

  - Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid lose of data.

  - All the data types of the variables are upgraded to the data type of the variable with largest data type.

- **bool -> char -> short int -> int -> unsigned int -> long -> unsigned -> long long -> float -> double -> long double**

EXAMPLE

```
#include<stdio.h>
int main()
{
    int x = 10;    // integer x
    char y = 'a';  // character c

    // y implicitly converted to int. ASCII
    // value of 'a' is 97
    x = x + y;

    // x is implicitly converted to float
    float z = x + 1.0;

    printf("x = %d, z = %f", x, z);
    return 0;
}
```

Output:

x = 107, z = 108.000000

- **Explicit Type Conversion**– This process is also called type casting and it is user defined. Here the user can type cast the result to make it of a particular data type.

- The syntax in C:

  (type) expression

Example

```
#include<stdio.h>

int main()
{
    double x = 1.2;

    // Explicit conversion from double to int
    int sum = (int)x + 1;

    printf("sum = %d", sum);

    return 0;
}
```

Output: sum = 2

SIZEOF OPERATOR

- Syntax: sizeof(object)
  - object can be of type (int, float etc)

- Returns the number of bytes needed to store an object.

```
printf("char: %u\n", sizeof(char));
printf("Int: %u\n", sizeof(int));
```

Math Library function

- **double ceil(double x):** returns the smallest integer value greater than or equal to x.

```
#include <stdio.h>
#include <math.h>
int main ()
{
float val1, val2;
val1 = 1.6;
val2 = -2.8;
printf ("value1 = %f\n", ceil(val1));
printf ("value2 = %f\n", ceil(val2));
return(0);
}
```

Output:

value1 = 2.0

value2 = -2.0

- **double floor(double x):** returns the largest integer value less than or equal to x.

```
#include <stdio.h>
#include <math.h>
int main ()
{
float val1, val2;
val1 = 1.6;
val2 = -2.8;
printf ("value1 = %f\n", floor(val1));
printf ("value2 = %f\n", floor(val2));
return(0);
}
```

Output:

value1 = 1.0

value2 = -3.0

- **double fabs(double x)**: returns the absolute value of x.

```
#include <stdio.h>
#include <math.h>
int main ()
{
   int a, b;
   a = 1234;
   b = -344;
   printf("The absolute value of %d is %d\n", a, (int)fabs(a));
   printf("The absolute value of %d is %d\n", b, (int)fabs(b));
   return(0);
}
```

Output:

The absolute value of 1234 is 1234

The absolute value of -344 is 344

- **double log(double x)**: returns the natural logarithm (base-e logarithm) of x.

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

int main ()
{
   double x, ret;
   x = 2.72;

   /* finding log(2.72) */
   ret = log(x);
   printf("log(%lf) = %lf", x, ret);

   return(0);
}
```

Output:

log(2.720000) = 1.000632

- **double log10(double x)**: returns the common logarithm (base-10 logarithm) of x.

```
#include <stdio.h>
#include <math.h>
 int main ()
{
   double x, ret;
   x = 10000;
   /* finding value of log1010000 */
   ret = log10(x);
   printf("log10(%lf) = %lf\n", x, ret);
   return(0);
}
```

Output:

log10(10000.000000) = 4.000000

- **double fmod(double x, double y)** : returns the remainder of x divided by y

```
#include <stdio.h>
#include <math.h>
int main ()
{
   float a, b;
   a = 8.2;
   b = 5.7;
   printf("Remainder of %f / %f is %lf\n", a, b, fmod(a, b));
   return(0);
}
```

Output:

Remainder of 8.200000 / 5.700000 is 2.500000

- **double sqrt(double x)**: returns the square root of x.

```
#include <stdio.h>
#include <math.h>
int main ()
{
   printf("Square root of %lf is %lf\n", 225.0, sqrt(225.0) );
   printf("Square root of %lf is %lf\n", 300.0, sqrt(300.0) );
   return(0);
}
```

Output:

Square root of 225.000000 is 15.000000

Square root of 300.000000 is 17.320508

- **double pow(double x, double y)**: returns x raised to the power of y i.e. x^y.

```
#include <stdio.h>
#include <math.h>
int main ()
{
   printf("Value 8.0 ^ 3 = %lf\n", pow(8.0, 3));
   printf("Value 3.05 ^ 1.98 = %lf", pow(3.05, 1.98));
   return(0);
}
```

Output:

Value 8.0 ^ 3 = 512.000000

Value 3.05 ^ 1.98 = 9.097324

- **double exp(double x)**: returns the value of e raised to the x^{th} power.

```
#include <stdio.h>
#include <math.h>
int main ()
{
   double x = 0;
   printf("The exponential value of %lf is %lf\n", x, exp(x));
   printf("The exponential value of %lf is %lf\n", x+1,
exp(x+1));
   return(0);
}
```

Output:

The exponential value of 0.000000 is 1.000000

The exponential value of 1.000000 is 2.718282

- **double cos(double x)** : returns the cosine of a radian angle x.

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159265
int main ()
{
   double x, ret, val;
    x = 60.0;
   val = PI / 180.0;
   ret = cos( x*val );
   printf("The cosine of %lf is %lf \n", x, ret);
   return(0);
}
```

Output:

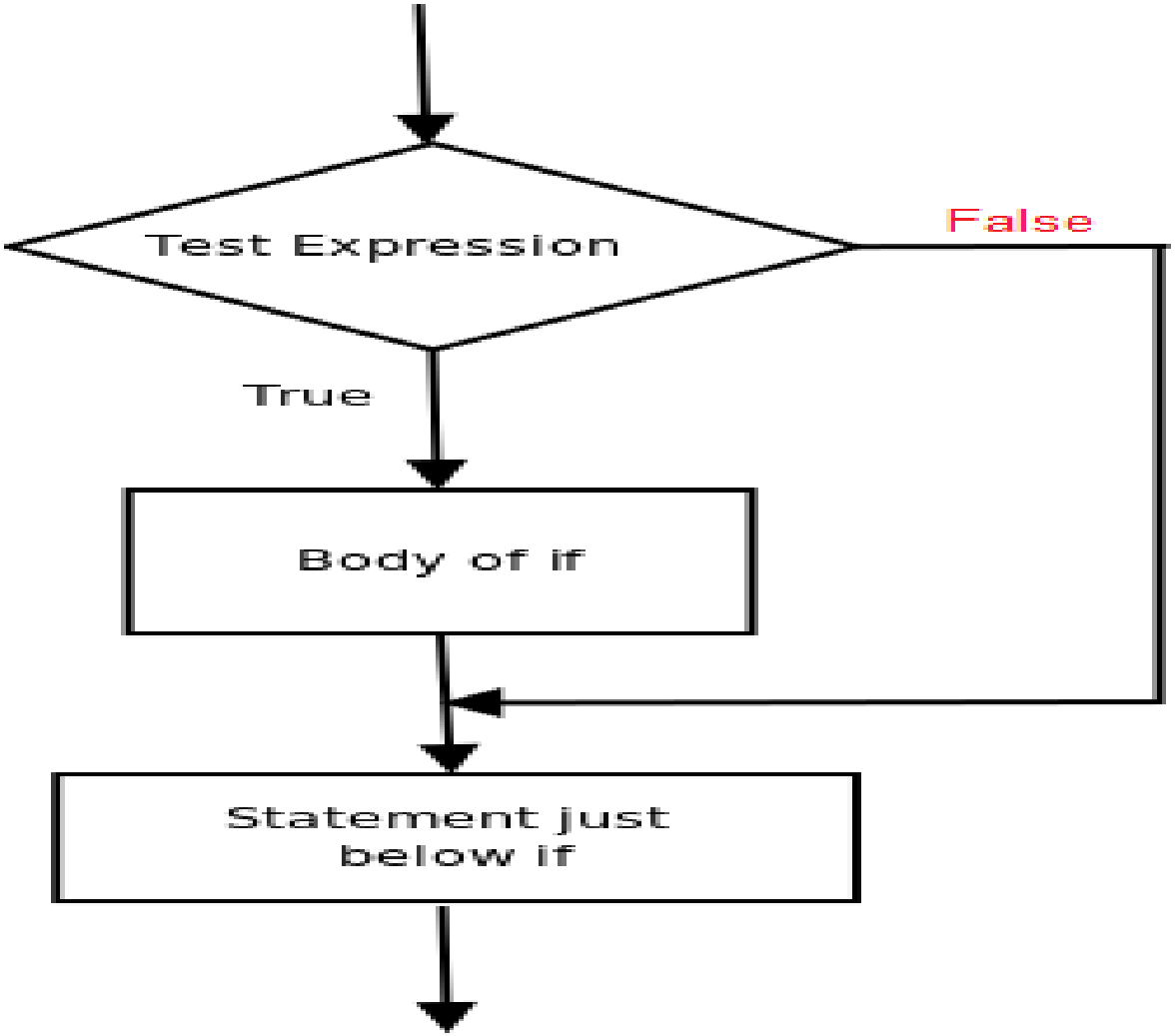
The cosine of 60.000000 is 0.500000

- Note on compilation with math library
- cc –lm program_name.c –o object_file_name

Flow Control - if

- if (expr) stmt
  - expr is an expression. If expr is nonzero (TRUE) then execute the statement stmt else skip it.
  - The statement stmt may be one statement or a compound statement
  - Usually expr will be a relational, equality or logical expression

Flowchart

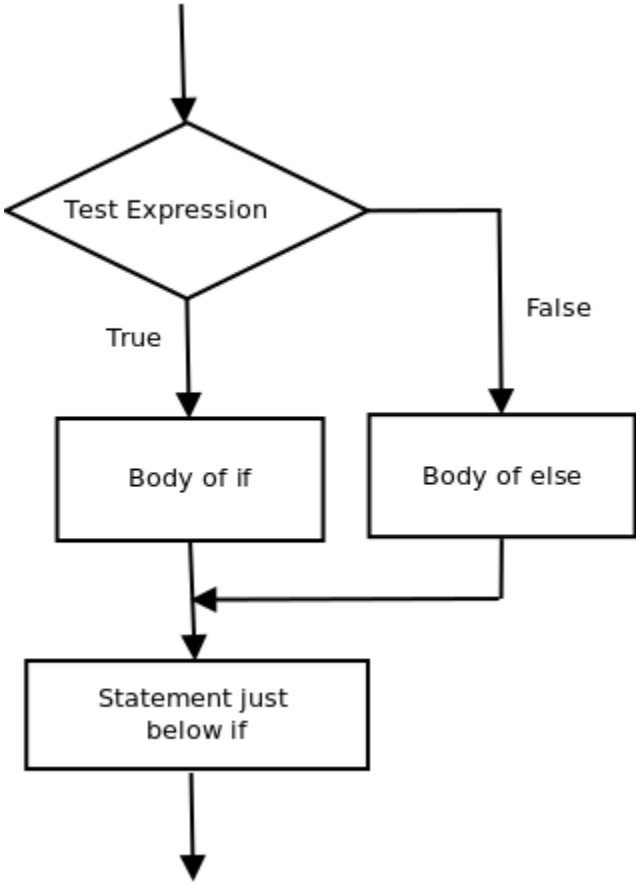

Test Expression
False
True
Body of if
Statement just
below if

Flow Control – if … else

```
if (expr)
      stmt1
 else
      stmt2
```

- If expr is nonzero than stmt1 is executed and stmt2 is skipped else stmt2 is executed and stmt1 is skipped

Flowchart


Test Expression
True
False
Body of if
Body of else
Statement just
below if

Conditional Operator (?:)

- Syntax: expr1 ? expr2 : expr3
- Precedence just above assignment ops.
- Right to left associated.

```
if (y < z )
x = y;
else
        x = z;
```

x = (y < z) ? y : z

Operators (7)

Operator Precedence and Associativity	
Operator	Associativity
() ++(postfix) --(postfix)	left to right
+(unary) -(unary) ++(prefix) --(prefix) !	right to left
* / %	left to right
+ -	left to right
< > <= >=	left to right
== !=	left to right
&&	left to right
||	left to right
?:	right to left
= += -= *= /= etc	right to left
, (comma operator)	left to right

Check whether Number is Odd or Even

```
#include<stdio.h>
int main() {
int num;
printf("Enter the Number : ");
scanf("%d",&num);
if(num%2==0)
    printf(“\Number is even\n”);
else
    printf(“Number is odd\n”);
return 0;
}
```

Assignment

- Take ages of two persons as input from user (might be floating point) and find if either of them is divisible by other
- Take an integer number as input and check if the input is a n digit decimal number
- Take an integer (below 10000) number as input and count the number of digit in it.

Take ages of two persons as input from user (might be floating point) and find if either of them is divisible by other

```
#include<stdio.h>
#include<math.h>
int main(){
float x,y;
printf("Enter value of x and y\n");
scanf("%f%f",&x,&y);
if(fmod(x,y)==0.0) printf("yes");
if (fmod(y,x)==0.0) printf("yes");
return 0;
}
```

Check if the input is a n digit decimal number

```
#include<stdio.h>
#include<math.h>
int main()
{
int num,n,limit_lower,limit_upper;
printf("Enter input number and \n");
scanf("%d%d",&num,&n);
limit_lower=pow(10,n-1);
limit_upper=pow(10,n) - 1;
if(num>=limit_lower && num <= limit_upper)
        printf("%d is a %d digit number\n",num,n);
else
        printf("%d is not a %d digit number\n",num,n);
return 0;
}
```

NUMBER AS INPUT AND COUNT THE NUMBER OF DIGIT IN IT.

```
#include<stdio.h>
int main(){
int num,flag=0;
printf("Enter the number\n");
scanf("%d",&num);
if(num<10) {printf("number is 1 digit"); flag=1;}
if (num>=10 && num <100) {printf("number is 2 digit"); flag=1;}
if (num>=100 && num <1000) {printf("number is 3 digit");flag=1;}
if (num>=1000 && num <10000) {printf("number is 4 digit");flag=1;)
If(flag==0) printf("Wrong input");

return 0;
}
```

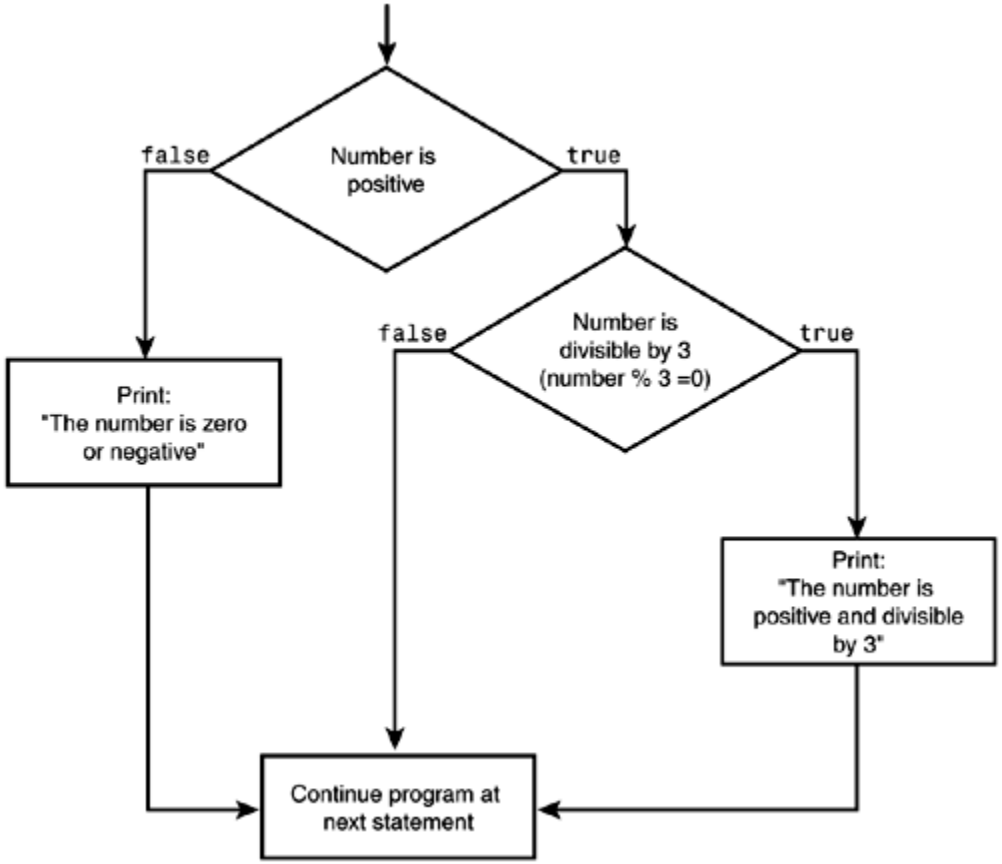
Nested If

1. if (expr) stmt ➔ stmt

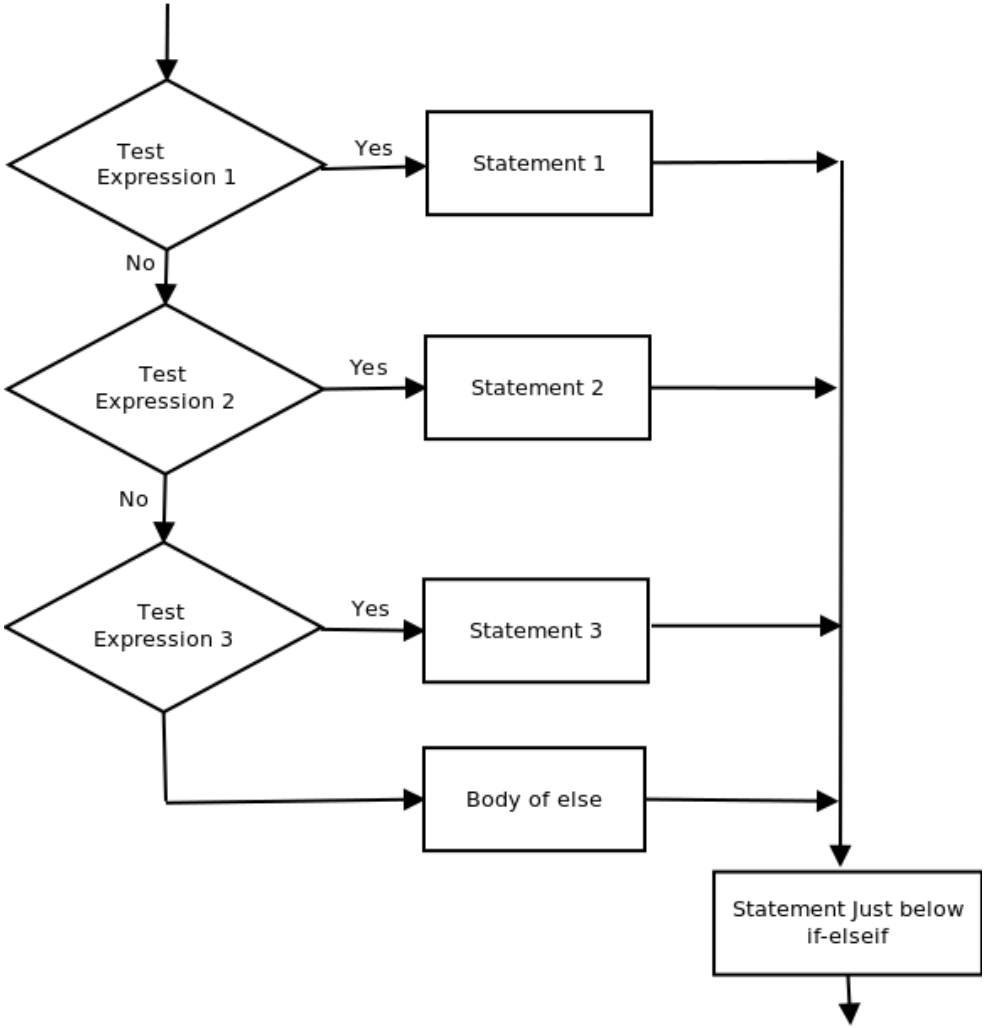
2.
```
if (expr)
        stmt1
else
        stmt2
```
➔ stmt

- Any stmt shown in 1 and 2 can be replaced by stmt
- For example if we replace stmt1 in 2 with 2 itself
-
```
if (expr)
        If(expr)
            stmt
        else
            stmt
else
        stmt
```

Nested if Flowchart



If-else-if ladder



- Take an integer as input and check if the input is positive and divisible by 3.

```
#include<stdio.h>
int main(){
int x;
printf("Enter value of x\n");
scanf("%d",&x);
if(x>=0)
        if(x%3==0)
                printf("Yes");
return 0;
}
```

Assignment using if else

- A student receives AA grade when he/she gets number in the range of 91-100, similarly AB for 81-90, BB for 71-80, BC for 61-70, CC for 51-60, CD for 41-50, DD for 31-40 and F if he/she scores less than or equal to 30. Write a program which will read the marks received as input and print corresponding grade.

- Bank offers loan at 15% interest rate. However, it offers 2% additional rebate if anyone is from rural area, 2% additional rebate if they belong to below poverty level class, and 2% additional rebate if it is taken for agriculture/fishery/dairy. Read the details of a person and compute the final interest rate.

SWITCH-CASE

- The if..else..if ladder allows you to execute a block code among many alternatives. If you are checking on the value of a single variable in if...else...if, it is better to use switch statement.

- syntax of switch statement is cleaner and easy to understand.

- Good for menu driven utility implementation

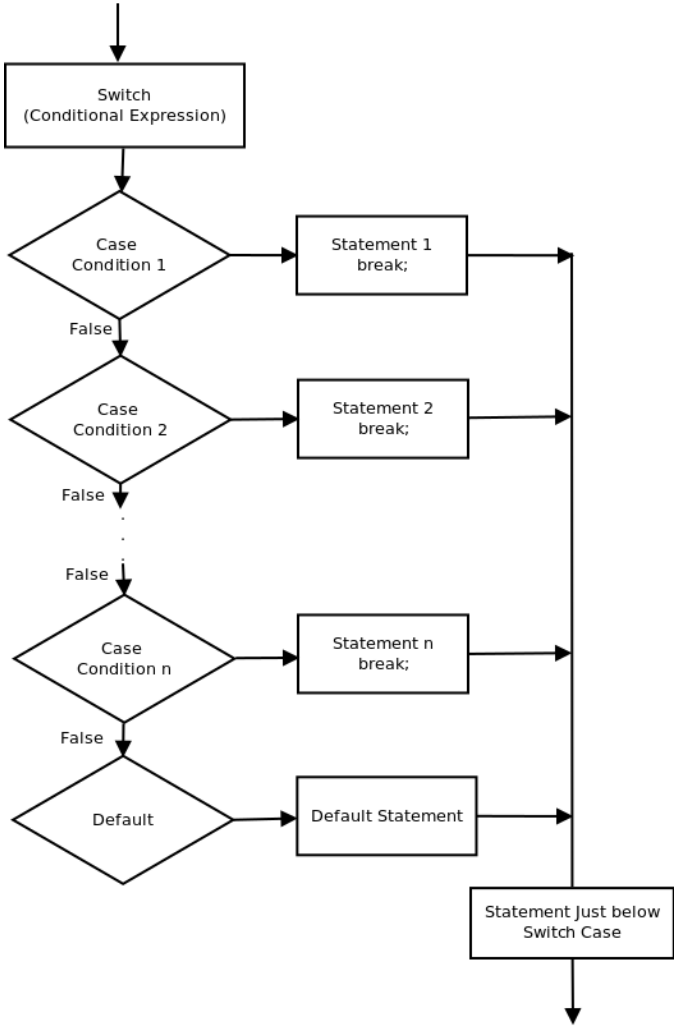
Switch Structure

switch (n) {

- case constant1:
  - // code to be executed if n is equal to constant1; break;
- case constant2:
  - // code to be executed if n is equal to constant2; break;
- default:
  - // code to be executed if n doesn't match any constant

}

Switch-case flowchart



Example – switch statement

```
……….
printf("Pick a number between 1 and 2\n");
scanf("%d", &a);
switch (a) {
     case 1: printf("You chose number 1\n");
              break;
    case 2: printf("You chose number 2\n");
              break;
   default: printf("That's neither 1 nor 2!\n");
    }
```

Range in switch case

- You can specify a range of consecutive values in a single case label, like this
  - **case low ... high:**
- It can be used for ranges of ASCII character codes like this
  - **case 'A' ... 'Z':**
- You need to Write spaces around the ellipses **...** . For example
  - // Correct - **case 1 ... 5:**
  - // Wrong - **case 1...5:**

Assignment using switch case

- A student receives AA grade when he/she gets number in the range of 91-100, similarly AB for 81-90, BB for 71-80, BC for 61-70, CC for 51-60, CD for 41-50, DD for 31-40 and F if he/she scores less than or equal to 30. Write a program which will read the marks received as input and print corresponding grade.

Importance of break in switch case

```
int main(){
        int n;
        printf("Enter your score\n");
        scanf("%d",&n);
        switch(n){
                case 1 ... 29: printf("Fail\n");
                case 30 ... 60: printf("Pass\n");
                case 61 ... 100: printf("Excellent\n");
        }
        return 0;
}
```

- [sourav@pg CS102]$ ./a.out
- Enter your score
- 5
- Fail
- Pass
- Excellent
- [sourav@pg CS102]$ ./a.out
- Enter your score
- 40
- Pass
- Excellent
- [sourav@pg CS102]$ ./a.out
- Enter your score
- 65
- Excellent

Importance of break in switch case

```
int main(){
        int n;
        printf("Enter your score\n");
        scanf("%d",&n);
        switch(n){
                case 1 ... 29: printf("Fail\n");break;
                case 30 ... 60: printf("Pass\n");break;
                case 61 ... 100: printf("Excellent\n");
                                 break;
        }
        return 0;
}
```

- [sourav@pg CS102]$ ./a.out
- Enter your score
- 5
- Fail
- [sourav@pg CS102]$ ./a.out
- Enter your score
- 40
- Pass
- [sourav@pg CS102]$ ./a.out
- Enter your score
- 65
- Excellent