# Unit-2

**Course Name: BCA**

**Subject Code : 1CS1010101**

**Subject Name: FUNDAMENTALS OF COMPUTER PROGRAMMING**

**Created By: Dr. Ronak Patel, I/C Principal, Shri C J Patel College of Computer Studies, Sankalchand Patel University,Visnagar.**

# Character set

- Four types of character set are used in C program.

**(1)Letters:-**

Upper case :→ A………Z

Lower case :→ a………z

**(2)Digits:-**

All decimal digits :→ 0……9

**(3) Special characters:-**

| | | | |
|---|---|---|---|
| , | comma | & | ampersand |
| . | period | ^ | caret |
| ; | semicolon | * | asterisk |
| : | colon | - | minus sign |
| ? | question mark | + | plus sign |
| ‘ | apostrophe | < | less than |
| “ | quotation mark | > | greater than |
| ! | exclamation mark | ( | left parenthesis |
| \| | vertical bar | ) | right parenthesis |
| / | slash | [ | left bracket |
| \ | back slash | ] | right bracket |
| ~ | tilde | { | left brace |
| _ | under score | } | right brace |
| $ | dollar sign | # | number sign |
| % | percent sign | | |

## Character set

**(4) White spaces:-**

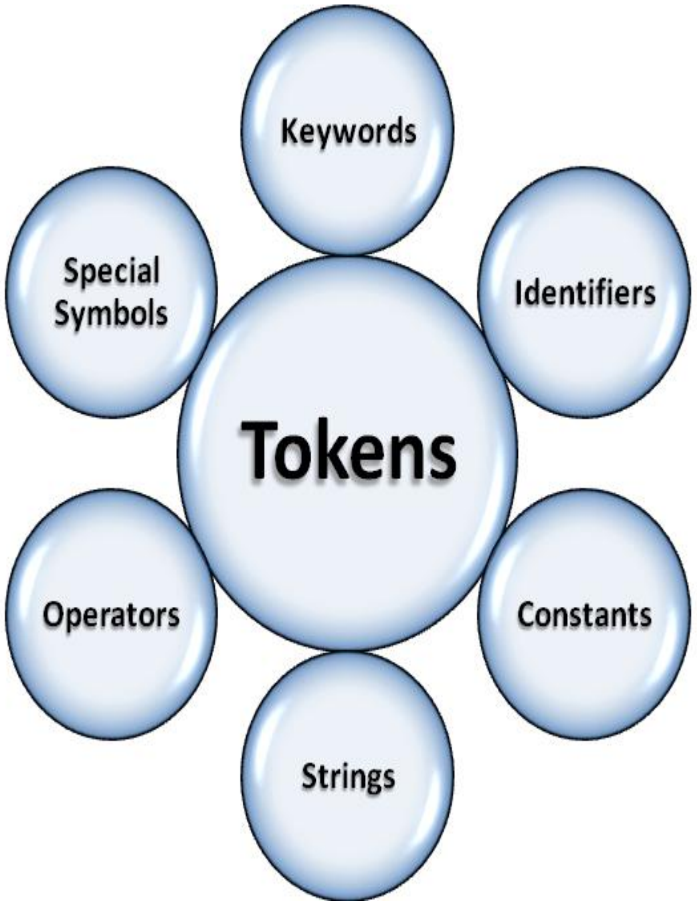
Blank spaces
Horizontal spaces
Carriage return
New line
Form feed

# C tokens

C Token: The smallest individual units of C Language is called C tokens.

There are six types of tokens are used in a C language.

1. Key-word
2. Identifiers
3. Constants
4. Strings
5. Operators
6. Special symbols



## Keywords

- Keywords means c language own words.
- Keywords are reserve words.
- Keywords are system defined words.
- Every keyword has some meanings that can’t be changed.
- Total 32 keywords are there in c language.

# 32 keywords

| **Keywords** | | | |
|---|---|---|---|
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| const | float | short | unsigned |
| continue | for | signed | void |
| default | go to | size of | while |
| do | if | static | Volatile |

## Identifier

- The name of any variable, array, function, string, pointer, structure, union etc is known as identifier.
- Identifier is a user defined.

e.g.

```
int x, y, sum;
```

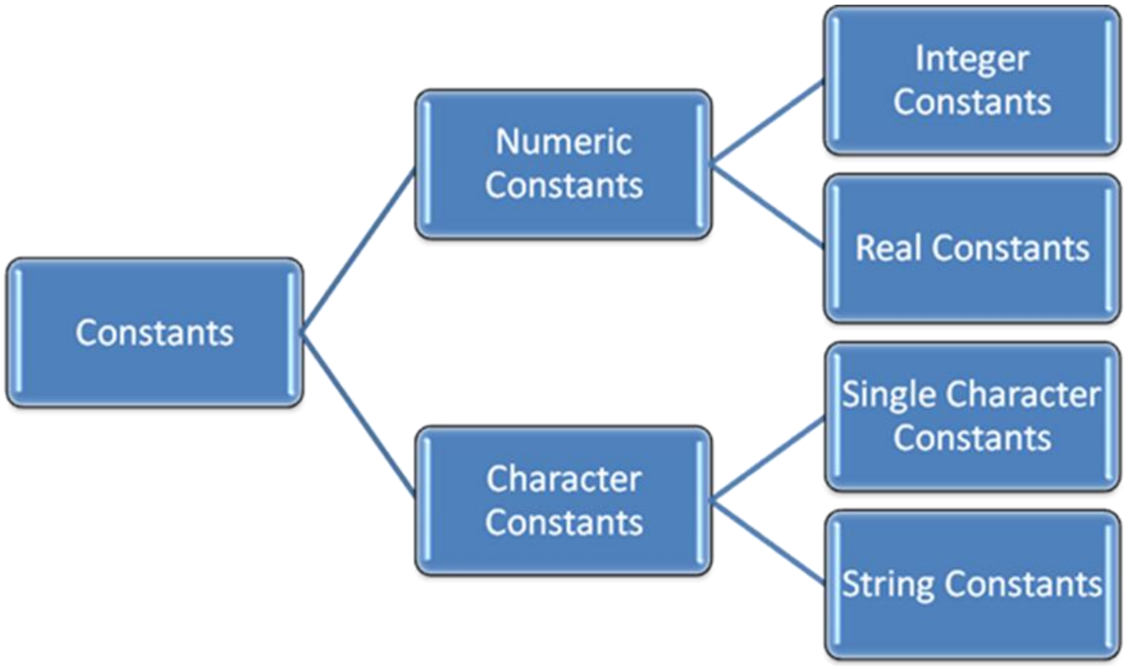
here, int is the keyword but x, y, sum are identifiers

## Naming rules of identifier Or Naming rules of variable

- Identifier made up of a to z letters ,0 to 9 digit and under score( _ )
- Under score( _ ) is allowed but white space is not allowed in identifier name.
- First character must be a letter.
- Keyword cannot be used as a identifier name.
- The length of identifier should not be more than 31 characters.
- Capital and small letters are significant. TOTAL and total are not same.

## Constant

- It is fixed value that does not change during the execution of program.
- C supports following constants…



## Integer constants

- A sequence of digits. There are three types…

  (1)Decimal integer

  - It is a set of digits, 0 through 9, with an optional sign + or -.
  - Valid Example: 123 -321 65432 +78

- (2)Octal integer

  - It is a set of digits, 0 through 7
  - Valid Example: 37 0 435 0551

  (3)Hexa-decimal integer

  - A sequence of digits (It is a set of 0 to 9 and A to F)
  - Valid Example: F2 57 50A A5

## Real Constants

- A sequence of digits with fractional part or decimal points.
- E.g   5.45, 3.14, 56.89, -67.89567

## Single Character Constants and String Constants

- **Single character constants:**
  - It contains a single character enclosed within a pair of single quote marks.
  - Single character constant have a equivalent ASCII value.
  - E.g ‘R’ ,‘c’, ‘a’, ‘ ’, ‘A’

- **String constants:**
  - It contains a sequence of characters enclosed within double quotes.
  - E.g. “Hello” , “RAM”, “1981”

# Sample Program
# (Program to implement the concept of constant)

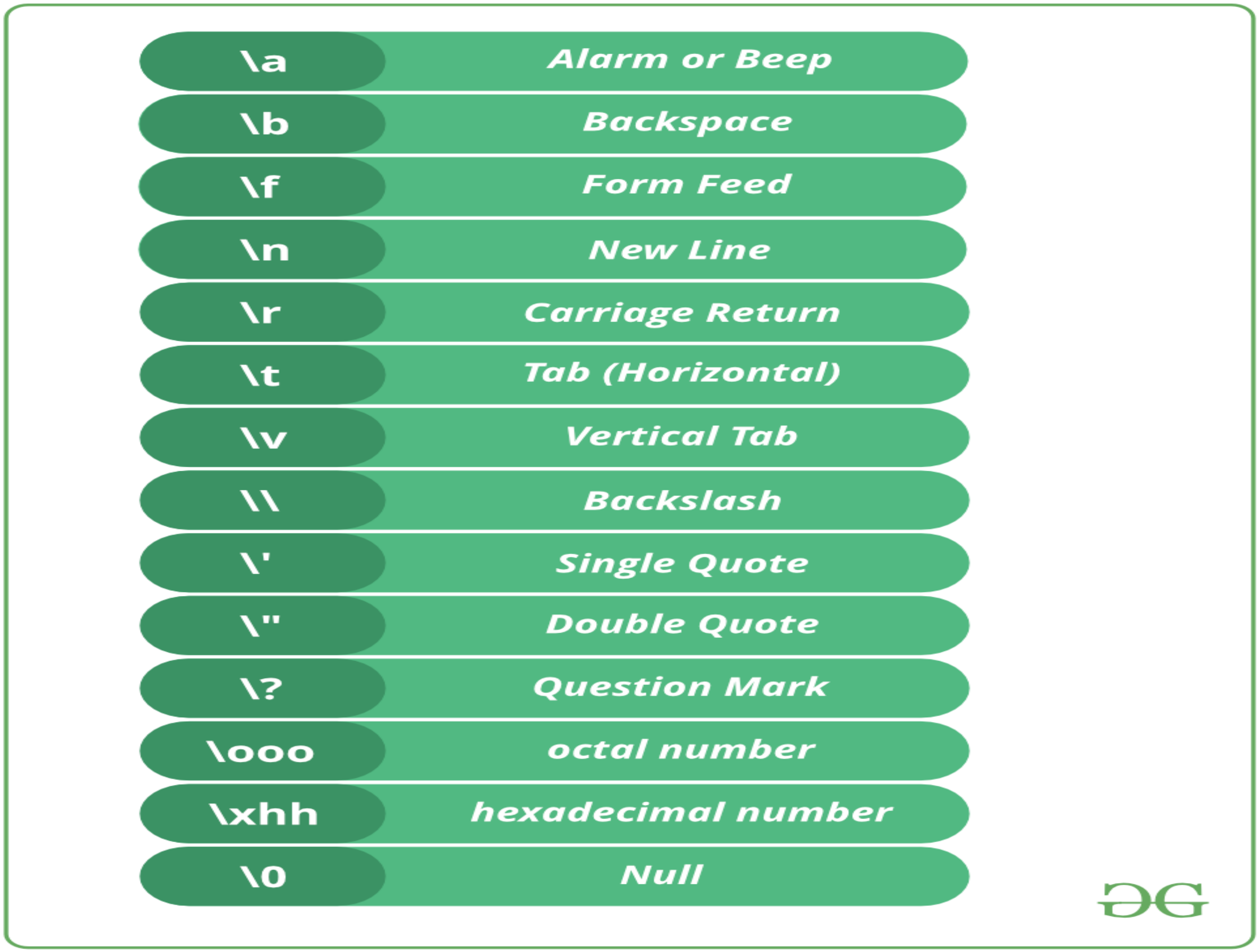
```
#include<stdio.h>
#include<conio.h>
void main(void)
{
 const int x=509;
 const float y=3.25;
 const char z='a';
 const char name[100]="RAM";
 clrscr();
 printf("Decimal integer constant=%d\n",x);
 printf("Real Constant=%f\n",y);
 printf("Single Character Constant=%c\n",z);
 printf("String Constant=%s\n",name);
 getch();
}
```

**Output:**

```
Decimal integer constant = 509
Real Constant = 3.250000
Single Character Constant = a
String Constant = RAM
```

## Backslash Character Constants

- It is used for better output of C program.
- It is also called Escape sequences.



| | |
|---|---|
| \a | Alarm or Beep |
| \b | Backspace |
| \f | Form Feed |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab (Horizontal) |
| \v | Vertical Tab |
| \\ | Backslash |
| \' | Single Quote |
| \" | Double Quote |
| \? | Question Mark |
| \ooo | octal number |
| \xhh | hexadecimal number |
| \0 | Null |

## Symbolic Constants

- The syntax is …

**#define Symbolic_Contant_Name Constant_Value**

- **E.g**

**#define PI 3.14**

**#define X 24**

## String

- String is a set of characters.
- It is collection of no. of characters.
- It is array of characters.
- E.g.  “Ram”,”Laxman”,”125”,etc

## Operators

- It is symbol which is used to solve an expression.
- E.g.   +,  -,  * ,  / ,  % ,< , > etc.

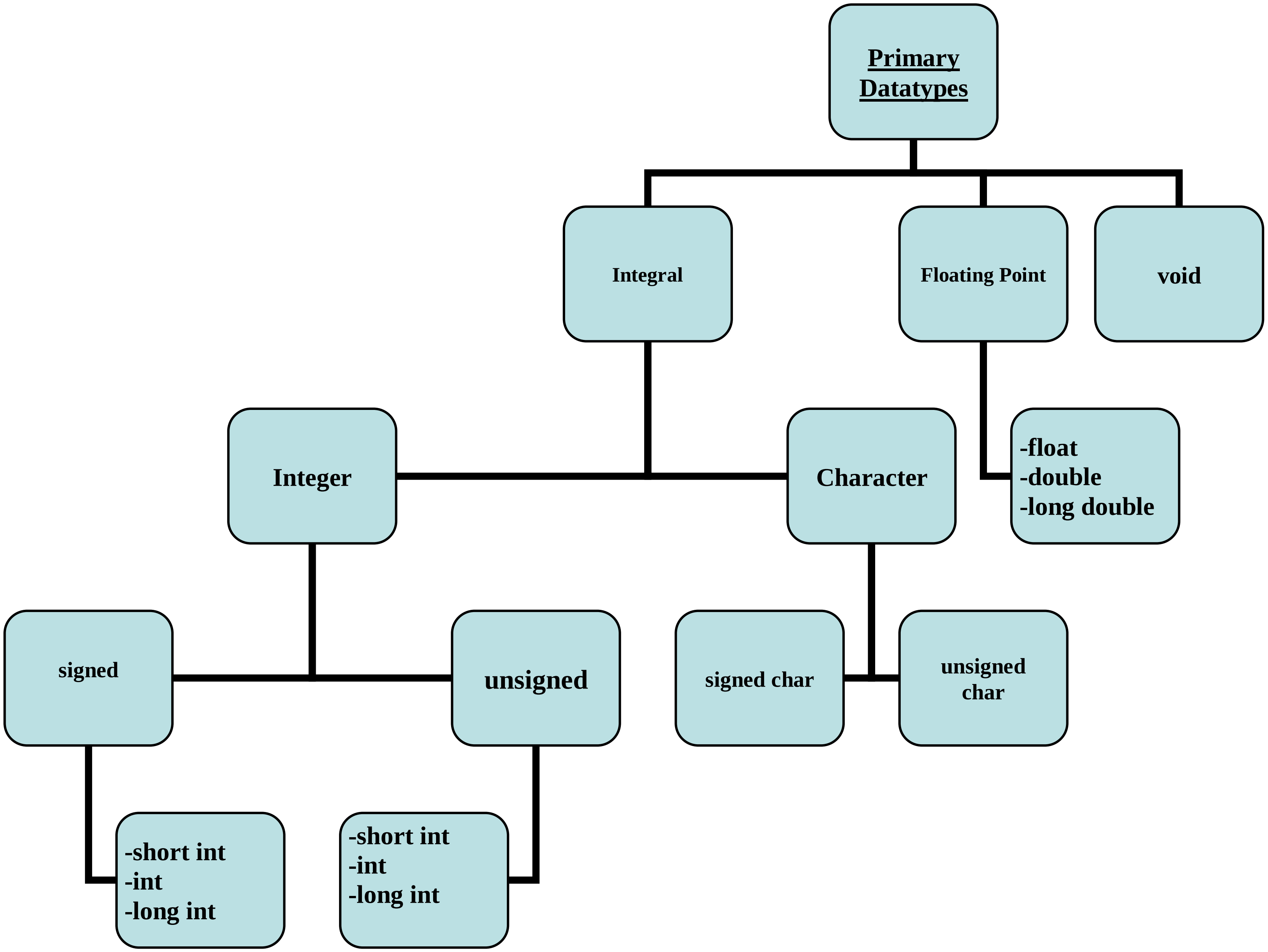
## 6. Special symbols

- {, }, [, ], ( , ), ; , .  etc are called special symbol.

## Data type

- **Datatype means type of variable.**
- **There are three types of datatypes used in C language.**
  1. **Primary datatype**
  2. **User device datatype**
  3. **Device datatype**

# Primary Datatype



## Primary Datatype

- Integer datayype used for storing integer value
- Character datatype is used for storing character value.
- Similarly Floating point datatype is used for storing a real value.
- Signed means it can store positive and negative both  types of values.
- While unsigned means it can store only positive value.

## Range of Integral types

| | **DataType** | **Bits** | **Range** |
|---|---|---|---|
| 1. | signed short int | 8 | -128 to 127 |
| 2. | unsigned short int | 8 | 0 to 255 |
| 3. | int or signed int | 16 | -32768 to 32767 |
| 4. | unsigned int | 16 | 0 to 65535 |
| 5. | signed long int | 32 | -2147483648 to 2147483647 |
| 6. | unsigned long int | 32 | 0 to 4294967295 |

## Range of Floating point types

| | DataType | Bits | Range |
|---|---|---|---|
| 1. | float | 32 | 3.4E – 38 to 3.4E + 38 |
| 2. | double | 64 | 1.7E – 308 to 1.7E + 308 |
| 3. | long louble | 80 | 3.4E – 4932 to 1.1E + 4932 |

## Format Specifiers for printf and scanf

| Data Type | Printf specifier | Scanf specifier |
|---|---|---|
| long double | %Lf | %Lf |
| double | %f | %lf |
| float | %f | %f |
| unsigned long int | %lu | %lu |
| long int | %ld | %ld |
| unsigned int | %u | %u |
| int | %d | %d |
| short | %hd | %hd |
| char | %c | %c |

## Sample Program
## (Program to implement the concept of Datatypes)

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
  short int a = -20;
  signed int b=30000;
  unsigned int c=50000;
  long int d=-100000;
  unsigned long int e=2000000;
  float f=35.5;
  double g=234566677.67886;
  long double h=345556.6788;
  char i='R';
  clrscr();
  printf("a=%hd\n",a);
  printf("b=%d\n",b);
  printf("c=%u\n",c);
  printf("d=%ld\n",d);
  printf("e=%lu\n",e);
  printf("f=%f\n",f);
  printf("g=%lf\n",g);
  printf("h=%Lf\n",h);
  printf("i=%c\n",i);
  getch();
}
```

**Output:**

```
 a=-20
b=30000
c=50000
d=-100000
e=2000000
f=0.000000
g=0.000000
h=345556.678800
i=R
```

## User define datatypes

C supports two kinds of user defined datatypes.

1. **typedef**
2. **enum**

## typedef

- Its refers to an existing data type with new identifier.
- The general syntax is as below.

  **typedef <data-type> <new_identifier>**

- typedef cannot create new data type, its represent existing data-type.
- **E.g**

```
typedef int marks;
```

They can be later to declare the variable as follows…

```
marks m1,m2;
```

## Sample Program
## (Program to implements the concept of typedef

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    typedef int marks;
    typedef float shape;
    marks m1,m2;
    shape s1,s2;
    clrscr();
    m1=70;
    m2=80;
    s1=45.234;
    s2=46.405;
    printf("Marks of sub1=%d\n",m1);
    printf("Marks of sub2=%d\n",m2);
    printf(“Shape1=%f\n",s1);
    printf(“Shape2=%f\n",s2);
    getch();
}
```

**Output:**

```
Marks of sub1=70
Marks of sub2=80
Shape1=45.234000
Shape2=46.405000
```

## Enumeration

- Another user defined data type is enumerated data type.
- General syntax is as below.

  **enum identifier {value1, value2,…valuen};**
- The values defined in braces that called enumeration constants.

# Sample Program
# (Program to implements the concept of enum)

```
#include<stdio.h>
#include<conio.h>
enum days
{
 mon=1,
 tue,
 wen,
 thu,
 fri,
 sat,
 sun
};
void main(void)
{
    enum days weekstart,weekend;
    clrscr();
    weekstart=mon;
    weekend=sun;
    printf("Week Start=%d\n",weekstart);
    printf("Week End=%d\n",weekend);
    getch();
}
```

**Output:**

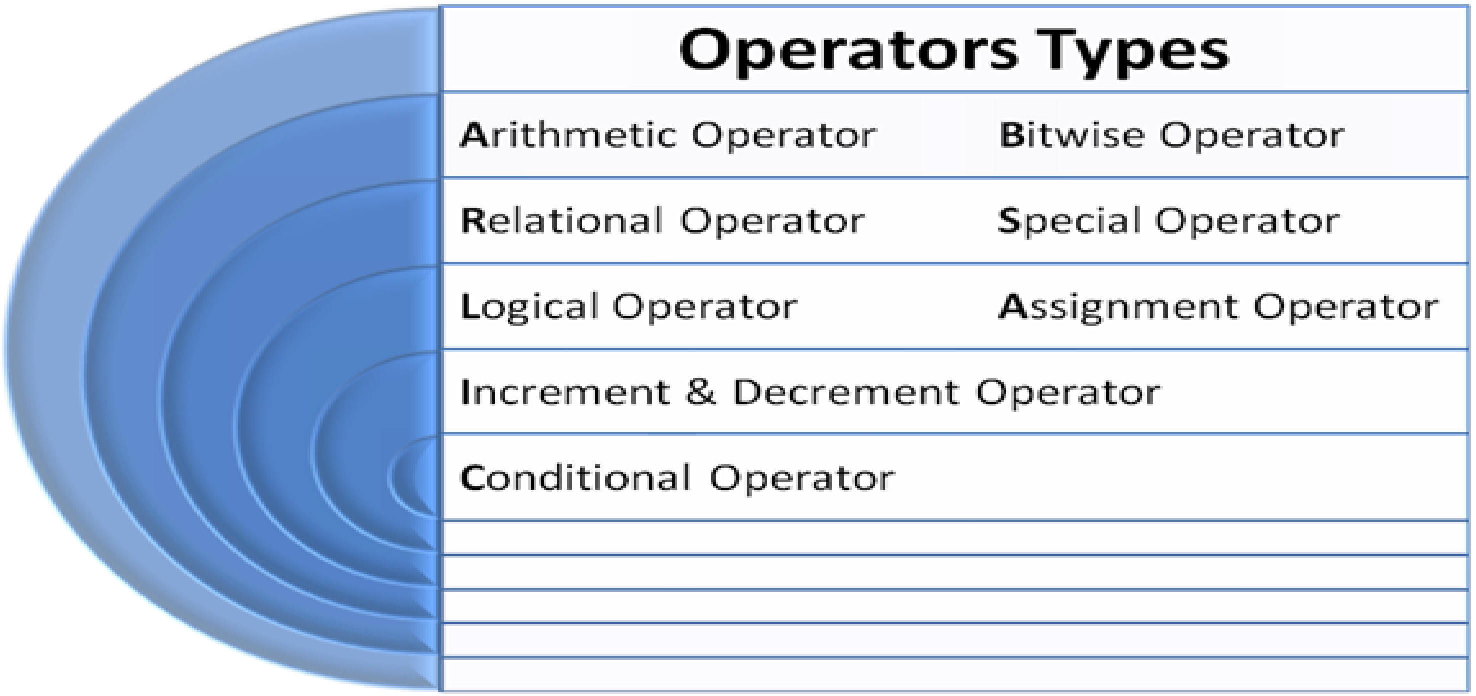
```
Week Start=1
Week End=7
```

## Derived Datatype

- **Derive datatype means it is derive from other datatypes.**
- **E.g. Array, String, Function, Pointer, structure , Union etc are the examples  of derived datatype.**

## Operators

- An **Operator** is a symbol that tells the computer to perform any operations
- There are 8 types of operators available in c-language.



| Operators Types | |
|---|---|
| Arithmetic Operator | Bitwise Operator |
| Relational Operator | Special Operator |
| Logical Operator | Assignment Operator |
| Increment & Decrement Operator | |
| Conditional Operator | |

## Arithmetic operator

- Arithmetic operators are used for arithmetic calculation.
- There are five arithmetic operators are available.
- **E.g.** A+B, A-B, A*B – where A and B are called operands and +, -, * are called operators.

| Operator | Meaning |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulo (Remainder)- Only for Integer |

# Arithmetic Expression

1. **Integer Arithmetic**
2. **Real Arithmetic**
3. **Mixed-mode Arithmetic**

## Integer Arithmetic Expression

- If both operands of arithmetic expressions are integer value, then this type of expression is called integer arithmetic expression.

  E.g.

```
int x , y, z;
    x=10;
    y=20;
    z=x + y;
```

- Here, x + y is a integer arithmetic expression.
- The result of integer arithmetic expression is always integer value.

## Real Arithmetic Expression

- If both operands of arithmetic expressions are real value then this type of expression is called real arithmetic expression.

E.g.

```
        float x, y, z;
            x=10.5;
            y=20.3;
            z = x + y;
```

- Here, x+y is a integer arithmetic expression.
- The result of real arithmetic expression is always real value.

## Mixed mode Arithmetic Expression

- If one operand is integer and other operand is real value,then this type of arithmetic expression is called mixed mode expression.

E.g.

```
          int x;
          float y , z;
           x=10;
           y=20.3
           z=x + y;
```

- Here, x + y is a mixed mode arithmetic expression.
- The result of mixed mode arithmetic expression is always real value.

## Sample program to implement Arithmetic operators

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
long int x,y,sum,sub,mul,div,rem;
clrscr();
printf("Enter any two values from keyboard\n");
scanf("%ld%ld",&x,&y);
sum=x+y;
sub=x-y;
mul=x*y;
div=x/y;
rem=x%y;
printf("Sum=%ld\n",sum);
printf("Sub=%ld\n",sub);
printf("Mul=%ld\n",mul);
printf("Div=%ld\n",div);
printf("Rem=%ld\n",rem);
getch();
}
```

**Output:**

```
Enter any two values from keyboard
50
100
Sum=150
Sub=-50
Mul=5000
Div=0
Rem=50
```

## Relational Operators

- C supports six relational operators.

| Operator | Meaning |
|---|---|
| < | is less than |
| < = | is less than or equal to |
| > | is greater than |
| > = | is greater than or equal to |
| = = | is equal to |
| ! = | is not equal to |

## Relational Operators

- The comparison is done with help of relational operator.
- relational operators are used for making a relational condition.
- relational condition has only two result it is either true or false.

E.g.

```
int  x , y;
 x=10;
y=20 ;
```

| **Relational Condition** | **Result** |
|---|---|
| x > y | False |
| x < y | True |
| x + y < 30 | False |
| x >= y | False |
| x <= y | True |
| x = = y | False |
| x! = y | False |

## Logical Operators

- There are three types of logical operators available in c language.

| Operator | Meaning |
|---|---|
| && | meaning logical AND |
| \|\| | meaning logical OR |
| ! | meaning logical NOT |

## Logical Operators

- It is used to join more than one relational conditions

| Condition 1 | Condition 2 | Condition1 && Condition2 | Condition1 \|\| Condition2 |
|---|---|---|---|
| T | T | T | T |
| T | F | F | T |
| F | T | F | T |
| F | F | F | F |

## Sample Program to implement Relational and Logical Operator

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
  int clang,office,co,cs,total,per;
  clrscr();
  printf("Enter the marks of four subjects\n");
  scanf("%d%d%d%d",&clang,&office,&co,&cs);
  total=clang+office+co+cs;
  per=total/4;
if(clang>=35&&office>=35&&co>=35&&cs>=35)
 {
        printf("Per=%d\n",per);
        if(per>=70)
        {
         printf("Distinction");
         }
         else if(per>=60 && per<70)
         {
          printf("First Class");
          }
         else
          {
           printf("Second class");
          }
  }
else
  {
  printf("Fail");
  }
getch();
}
```

**Output:**

```
Enter the marks of four subjects
50
60
70
80
Per=65
First Class
```

## Short Hand Assignment Operator

- Assignment operator is used to assign the result of an expression to a variable
- We can also assign value in short hand way.
- = is called shorthand assignment operator.
- The general syntax of shorthand assignment operators is as below.

**Variable op = exp;**

- **e.g**

| | |
|---|---|
| **a=a + 1** | **a + = 1** |
| **a =a - 1** | **a - = 1** |
| **a = a * (n+1)** | **a * = n + 1** |
| **a = a / (n+1)** | **a / = n + 1** |

## Sample program for Short hand Assignment operator

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
 int x=1,y=5,z=8;
 clrscr();
 x+=5;    //x=x+5;
 y-=y*3;  //y=y-(y*3)
 z%=7;    //z=z%7;
 x*=3;    //x=x*3;
 printf("x=%d\n",x);
 printf("y=%d\n",y);
 printf("z=%d\n",z);
 getch();
}
```

### Output:

```
x = 18
y = -10
z = 1
```

## Increment and Decrement Operators

- There are main two operators.
    - Increment Operator **+ +**
    - Decrement Operator **- -**
- Increment operator is used for increasing the value of variable by 1.
- Decrement operator is used for decresing the value of variable by 1.

- Increment operator has two types…
    - Pre Increment **+ + m;**
    - Post Increment **m + +;**

- Decrement operator has two types…
    - Pre Decrement **- - m;**
    - PostDecrement **m- -;**

## Sample program to implements Increment and Decrement Operator

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int p,q,r,s;
    clrscr();
    p=5;
    q=10;
    r=15;
    s=20;
    ++p;    //p=p+1;
    q++;     //q=q+1;
    --r;        //r=r-1;
    s--;      //s=s-1;
    printf("p=%d\n",p);
    printf("q=%d\n",q);
    printf("r=%d\n",r);
    printf("s=%d\n",s);
    getch();
}
```

**Output:**

```
p=6
q=11
r=14
s=19
```

## Sample program to implements Pre and Post Increment and Decrement Operators

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
   int p,q,r,s,z,k,y;
   clrscr();
   p=5;
   q=10;
   r=15;
   s=20;
   z=(++p)+10;       //6+10=16
   k=(q++)+15;     //10+15=25
   y=(--r)+(s--)+20;   //14+20+20=54
   printf("p=%d\n",p);
   printf("q=%d\n",q);
   printf("z=%d\n",z);
   printf("k=%d\n",k);
   printf("r=%d\n",r);
   printf("s=%d\n",s);
   printf("y=%d\n",y);
   getch();
}
```

**Output:**

```
p=6
q=11
z=16
k=25
r=14
s=19
y=54
```

## Conditional Operator(?)

- It is also called Turnery operator.
- The general syntax of conditional operators is as below.

  Variable = Exp1 ? Exp2 : Exp3 ;
- e.g.

```
 max = x >= y ? x : y;
it is similar to
if (x>=y)
{
max=x;
}
else
{
max=y;
}
```

## Sample program to implements Conditional Operator

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
    int max,x,y;
    clrscr();
    printf("Enter two values\n");
    scanf("%d%d",&x,&y);
    max= x>=y ? x : y;
    printf("max=%d\n",max);
   getch();
}
```

**Output:**

```
Enter two values
50
100
max=100
```

## Bitwise Operators

- Bitwise operator used for modify data at bit level.
- These operators are used for testing bits.
- Bitwise operators may not be applied to **float** and **double.**

| Operator name | Meaning |
|---|---|
| & | Bitwise AND |
| \| | Bitwise OR |
| ^ | Bitwise exclusive OR |
| << | Shift left |
| >> | Shift right |

## Special Operators

1. comma operator ( , )
2. size of operator ( sizeof() function)
3. member selection operator( . And -> )
4. pointer operator ( & and * )

## Sample program to implement Comma Operator(,)

```
#include<stdio.h>
#include<conio,h>
void main(void)
{
int x ,y ,z;
clrscr();
z=(x=10,y=20,x+y);
printf(“z=%d\n” , z);
getch();
}
```

**Output:**

```
z=30
```

## sizeof operator

- The sizeof operator is a compile time operator, when used with an operand, it returns the number of bytes the operand occupies.

For Example

**m=sizeof(sum);**

## Sample program to implements sizeof() Operator

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
 int a,b,c,d,e,f,g;
 int x;
 float y;
 char z;
 clrscr();
 a=sizeof(x);
 b=sizeof(y);
 c=sizeof(z);
 d=sizeof(int);
 e=sizeof(float);
 f=sizeof(char);
 g=sizeof(long double);

 printf("size of varible  x=%d\n",a);
 printf("size of varible  y=%d\n",b);
 printf("size of varible  z=%d\n",c);

 printf("size of int data type =%d\n",d);
 printf("size of float data type=%d\n",e);
 printf("size of char data type=%d\n",f);
 printf("size of long double data type=%d\n",g);
 getch();
}
```

**Output:**

```
size of varible x=2
size of varible y=4
size of varible z=1
size of int data type =2
size of float data type=4
size of char data type=1
size of long double data type=10
```