String

String :- A string is a sequence of character which is terminated by delimiter ('\0') and is enclosed in dobble qotation marks(" ") string is also known as character of Array.

Declaration of string :-

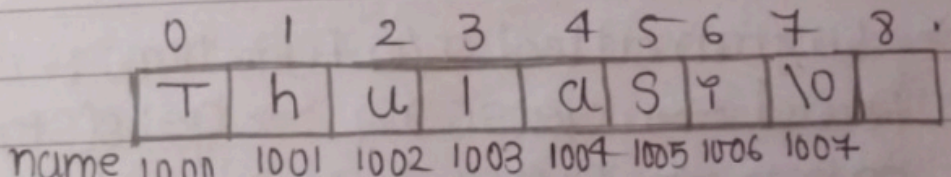
Syntax :- char stringname[size];

ex :- char name[20];

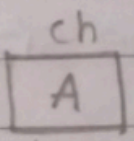
Initialization :-

char name[20] = "Thulasi";

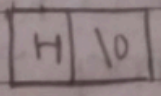
chare name[20] = {'T','h','u','l','a','s','i','\0'};



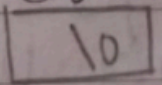
char ch = 'A'



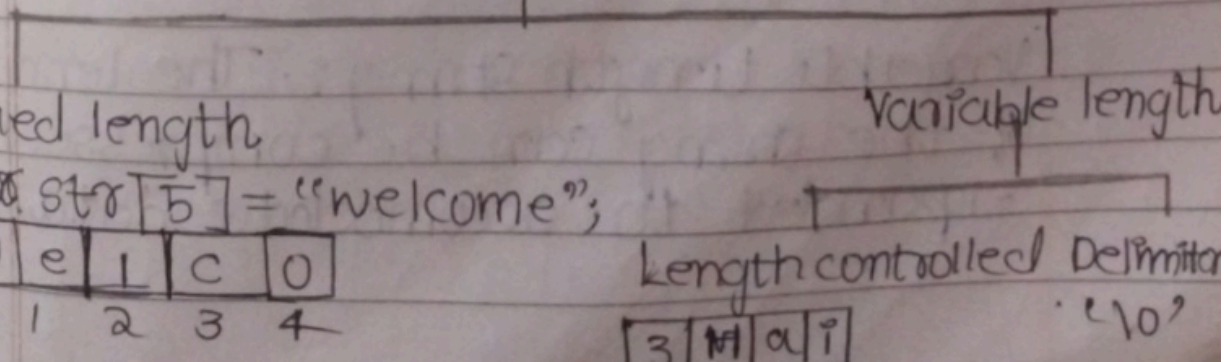
char ch = "H"



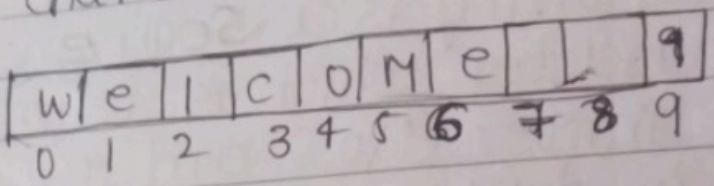
char ch = "";



String classifications



char str[10] = "welcome";



Variable length
1. length controlled :- By giving string the size memory will be wasted

String Input/output Functions :-

➡ Format specifier for string is %s
→ Formatted I/O functions for string
scanf("%s", str);
printf("%s", str);

→ unformatted I/O functions :-
Read → gets() → gets(str);
print → puts() → puts(str);

Fixed length string :- The size of string is fixed
disadvantages :- If we take to small size string we can't store it. If we take it as two big than wastage of memory.

Variable length string :- The length of the string can be compressed or expanded to accomodate data.

Length stored string :- These strings added a count which specifies the number of characters in the string at the starting itself.

Delimitter string :- This string end with delimitter at the end of the string the most common delimitter is null character '\0'.
The ascii value of null character is zero.

→ String is also known as character of Array
→ No need of for using loop because of %s
→ No need to place & in scanf it will automally takes base Address.
→ There is a disadvantage of scanf if we write space like " %s " it will take a null character '\0'.
→ gets(str) takes more and more variables until we enter press the enter button.

Q. write a c program to read and print string using formatted Input / output function.

Formatted :-

```
#include<stdio.h>
int main()
{
    char name[20];
    printf("enter string:\n");
    scanf("%s", name);
    printf("The string is: %s", name);
}
```

ouput :-

```
enter string:
Hello world
The string is:
Hello
```

unformatted :-

```
#include<stdio.h>
int main()
{
  char name[20];
  printf("enter string:\n");
  gets(name);
  puts(name);
}
```

output :-

```
Enter string:
   Hello world
Hello world
```

getchar(); I/p
putchar(); O/P

Character of Strings :-

```
char str[5][10] = {"Delhi", "Mumbai",
                   "Hyderabad", "Jaipur",
                   "Kolkata"};
```

	0	1	2	3	4	5	6	7	8	9
0	D	e	l	h	i	\0				
1	M	u	M	b	a	i	\0			
2	H	y	d	e	r	a	b	a	d	\0
3	J	a	i	p	u	r	\0			
4	K	o	l	K	a	t	a	\0		

A - 65
a - 97

Date
Page

character library function/ character input/output functions:-

Input - getchar()
output - putchar()
Header file is → #include<ctype.h>

Syntax:-

isalpha()	int isalpha(char c);
is digit()	int isdigit(char c);
isupper()	int isupper(char c);
is lower()	int islower(char c);
tolower()	char tolower(char c);
toupper()	char toupper(char c);

Description:-

1. If the given character is alphabetic, it returns true(1) else returns false(0)
2. If the given character is digit, it returns true(1) else returns false(0)
3. If the given character is upper, it returns true(1) else returns false(0)
4. If the given character is lower, it returns true (1) else (0)
5. It converts the given upper case character to lower case
6. It converts the given lower case character to upper case.

Example:-
isalpha(e) → returns 1
isalpha(9) → returns 0

isdigit(2) → returns 1
isdigit(a) → returns 0

isupper(G) → returns 1
isupper(g) → returns 0

islower(i) → returns 1
islower(I) → returns 0

tolower(J) → j
tolower(j) → j

toupper(H) → H
toupper(H) → (H)

* String handling functions (or) string manipulation function :-

→ To perform some operations on string we can use string handling functions.
→ To use string handling function we include the header file is
#include<string.h>

1. string length :- strlen()

→ This function is used to find the length of string.
→ It excludes the null character.
→ syntax :- int strlen(char s[20]);
size

ex :- char s[20] = "Hello world";
int l = strlen(s);
length = 11

Q. write a c program to find the length of string using strlen().

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s[20]; int l;
    printf("enter string:");
    gets(s);
    l = strlen(s);
    printf("length of string is: %d", l);
}
```

Q. write a c program to find the length of string without using strlen().

```
#include<stdio.h>
int main()
{
  char s[20], int i, c=0;
  printf("enter string:");
   gets(s);
  for(i=0; s[i]!='\0'; i++)
     c++;
  printf("length of string=%d\n",c);
}
```

H	e	l	l	o	\0
0	1	2	3	4	5

Length = 5

String comparison function :-

→ returns integer value.

→ string comparison function is used

→ to compare two strings

→ The two string comparison functions are

1. strcmp()
2. strncmp()

1. strcmp()

Syntax:-

```
int strcmp(char s1[20], char s2[20]);
```

2. strncmp()

```
int strncmp(char s1[20], char s2[20], int
```

n → Represents specified no. of characters

→ If both strings are equal it returns '0'.
→ If string1 is less than string 2 it returns -ve
→ If string1 is greater than string 2 it returns +ve

Str1 = Str2 → 0
Str1 < Str2 → -ve
Str1 > Str2 → +ve

Ex: Str1 = "Hello"; } returns 0.
Str2 = "Hello";

Str1 = "Abcd" A = 65 65-97 = -
Str2 = "abcd" a = 97
65 < 97 → returns -ve

Str1 = "abcd"; A = 65 a = 97
Str2 = "Abcd" 97 - 65 = +
97 > 65 → returns +ve

(8.) write a c program to check wheather two strings are equal or not using strcmp function.

```
# include<stdio.h
# include<string.h>
int main()
{
    char s1[20], s2[20];
```

```
int d;
printf("Enter string 1:");
gets(s1);
printf("Enter string 2:");
gets(s2);
d = strcmp(s1,s2)
if(d==0)
printf("Two strings are equal");
else if(d<0)
printf("string 2 is greater");
else (d>0)
printf("string 1 is greater:");
}
```

Q. Write a c program to check wheather the two strings are equal or not without using strcmp()

```
#include<stdio.h>
int main()
{
    char s1[20], s2[20];
     int l1, l2, i, flag=0;
    printf("Enter string1:");
    gets(s1);
    printf("enter string 2:");
    gets(s2);
    l1 = strlen(s1);
```

```
for(i=0; i<l1 || i<l2, i++)
{
    if(s1[i] != s2[i])
    {
        flag=1;
        break;
    }
}
if (flag==0)
printf("Two strings are equal");
else
printf("Two strings are not equal");
}
```

String concatenation:-

→ String concatenation function is used to combine one string to the end of another string.
→ The function returns the Address pointer to the destination string.
→ These are two types
1. Strcat()
2. Strncat()

1. Strcat()

Syntax:- strcat(char s1[20], char s2[20]);
Ex: str1 = "concate";
str2 = "nation";

Str1 = "concatenation";

2. strncat()

Strncat(char s1[20], char s2[20], int n);

Ex:-

```
Str1 = " wel";
str2 = " come";
Strncat(Str1, Str2, 3);
Str1 = " welcom";
```

(9) write a c program to concatenate two strings using strcat().

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[20], s2[20];
    printf("enter string1:");
    gets(s1);
    printf("enter string2:");
    gets(s2);
    strcat(s1, s2);
    puts(s1);
}
```

(10) write a c program to concatenate two strings without using strcat()

```
#include<stdio.h>
#include<string.h>
int main()
{
  char s1[20], s2[20]; int i, l1, l2;
  printf("Enter string 1:");
  gets(s1);
  printf("Enter string 2:");
  gets(s2);
  l1 = strlen(s1);
  l2 = strlen(s2);
  for(i=0; i<l2; i++)
  {
    s1[l1++] = s2[i];
  }
  s1[l1] = '\0';
  puts(s1);
}
```

s1 | w | o | r | l | d | \0 |

s2 | H | e | l | l | o | \0 |

s1 | w | o | r | l | d | h | e | l | l | o | \0 |

String copy function :-

→ String copy function is used to copy one string content to another string

→ these are of two types

1. strcpy()
2. strncpy()

1. strcpy()
Syntax:-
strcpy(char destination string[20], char sourstring[20]);
*destingstring
*Source string

ex:-
str1="Hai"
strcpy(str2, str1);
str2="Hai";

2. strncpy()
syntax:-
strncpy(char destinationstring[20], char sourstring[20], int n);

ex:-
str1="Hai"
strcpy(str2, str1, 2);
str2="Ha";

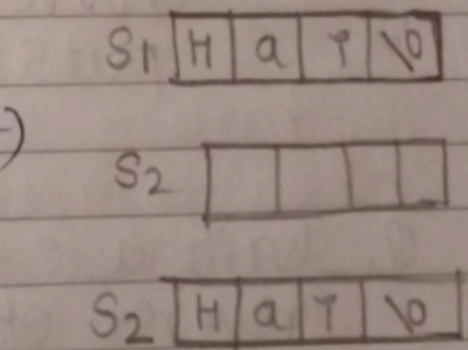
Q. Write a c program to copy the string one to another string using strcpy()

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s1[20], s2[20];
    printf("Enter string1:");
    gets(s1);
    strcpy(s2, s1);
    puts(s2);
}
```

Date
Page

(9) write a c program to copy the string one to another without using strcpy.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char s1[20], s2[20]; int l1, i;
 printf("enter string:");
  gets(s1);
 l1 = strlen(s1);
 for(i=0; i<l1; i++)
  {
    s2[i] = s1[i];
  }
  s2[i] = '\0';
  puts(s2);
}
```



String reverse :- strrev()

→ Strrev() function is used to reverse the given string.

Syntax :- strrev(char stringname[size]);

ex :- str1 = "Hello";
strrev(str1);
str1 = "olleH";

Q. write a c program to print the given string in reverse order using strrev().

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s[20];
    printf("enter string:");
    gets(s);
    strrev(s);
    puts(s);
}
```

Q. Write a c program to find the reverse of string without using strrev().

```
#include<stdio.h>
int main()
{
    char s[20];
    int i, l;
    printf("enter string:");
    gets(s);
    l = strlen(s);
    for(i=l-1; i>=0; i--)
    printf("%c", s[i]);
        (or)
// putchar(s[i]);//
}
```

9. write a c program to check whether the given string is palindrome or not.

```
# include<stdio.h>
# include<string.h>
int main()
{
   char s1[20], s2[20];
  printf("enter string:");
   gets(s1);
   strcpy(s2,s1);
   strrev(s1);
   if (strcmp(s1,s2)==0)
     printf("palindrome");
   else
     printf("not a palindrome");
}
```

8. write a program to replace lower case character to upper case & upper to lower characters.

```
#include<stdio.h>

int main()
{
   char s[20];
    int i;
```

A=65 A to a add +32
a=97 a to A sub -32

```
printf("enter string?");
gets(S);
for(i=0; S[i]!='\0'; i++)
{
    if(S[i]>='A' && S[i]<='Z')
      S[i]=S[i]+32;
    else if(S[i]>='a' && S[i]>='z')
      S[i]=S[i]-32;
}   else S[i]=' ';
puts(S);
}
```

Q. Write a C program delet n character in a given string based on specified position.

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

n=3

p	r	ø	g	r	a	m	\0
0	1	2	3	4	5	6	7

strlen=7 pos=2
n=3
i=2, j=2+3, 5<7
S[2]=S[5]
i=3, j=6, 6<7
S[3]=S[6]
i=4, 7<7 X

p	r	a	m	\0

```
#include<stdio.h>
#include<string.h>
int main()
{
    char s[20];
    int i, j, l, n, pos;
    printf("enter string:");
    gets(s);
    l = strlen(s);
    printf("enter position");
    scanf("%d", &pos);
    printf("enter number of characters
            to be deleted:");
    scanf("%d", &n);
    for(i=pos, j=pos+n; j<l; i++, j++)
        s[i] = s[j];
        s[j] = '\0';
        puts(s);
}
```

Substring → strstr()

→ This function is used to check sub string is present in mainstring or not

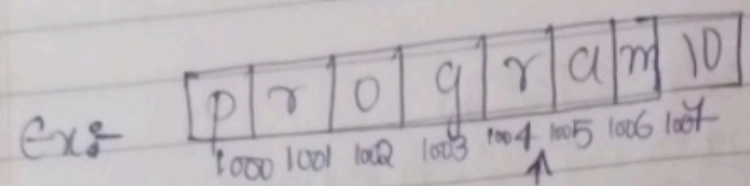
→ This return the position of sub string in the given mainstring

→ It returns the pointer to the substring.

Syntax:-

```
char *strstr(char mainstring[],
             char substring[]);
```

Ex:-




```
char *pos
pos = 1004      ms = 1000
= pos - ms + 1
= 1004 - 1000 + 1
= 4 + 1
= 5   (%d)
```

Q. write a c program to check substring is present in mainstring or not if substring present return the position otherwise return -1.

```
#include<stdio.h>
#include<string.h>
int main()
{
   char ms[20], ss[20], *pos;
   printf("enter main string:");
   gets(ms);
   printf("enter substring:");
   gets(ss);
   pos = strstr(ms, ss);
   if(pos)
```

printf("%s is present in %s at %d position:", ss, ms, pos-ms+1);
else
return -1;
}

strchr():-
→ It returns the position of given character in the given string from the beginning of the string.
Syntax:- char *strchr(char stringname[], char ch);
ex:- str = "program"
strchr(str, 'r');

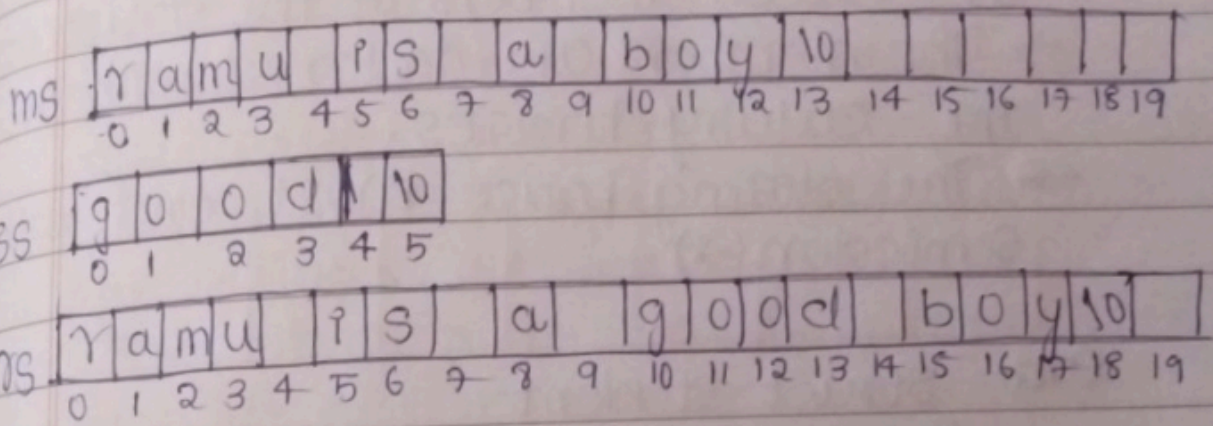
strrchr():-
syntax:-
char *strrchr(char stringname[], char ch);
ex:- str = "program"

Strrchr(str, 'r');
→ It returns the position of given character in the given string from the ending of the string.

write a program to insert sub-string into main string.

```
#include<stdio.h>
#include<string.h>
int main()
{
  int pos, ml, sl;
  char ms[30], ss[30], rs[30];
  printf("enter main string:");
  gets(ms);
  printf("enter substring:");
  gets(ss);
  printf("enter position:");
  scanf("%d", &pos);
  ml = strlen(ms);
  sl = strlen(ss);
  if(pos>ml)
  {
    for(i=0; i<sl; i++)
    ms[ml++]=ss[i];
    ms[ml]='\0';
    puts(ms);
  }
  else
  {
   for(i=0; i<pos; i++)
    rs[i]=ms[i];
    for(j=0; j<sl; j++)
    rs[i++]=ss[j];
   for(j=pos; j<ml; j++)
    rs[i++]=ms[j];
```

```
    rs[i] = '\0';
    puts(rs);
  }
}
```



Structure :- Structure is a collection of dissimilar (heterogenous) datatems stored in a continuous memory locations under a single name.

Structure can be declared in two ways:-
1. Tagged structure
2. Type definition structure.

1. Tagged structure

Syntax :-

```
struct Tag_name
{
    datatype member1;
    datatype member2;      > structure member.
    :
};
```

→ where struct is the keyword which tells the compiler that a structure is being defined.
→ Tag_name is a name of the structure
→ member1, member2,.... membern are called members of the structure.
→ The members are declared with in the culary braces.
→ The closing brace must end with semicolon(;)

ex:- struct student
{
char sname[20];
char rno[15];
int m1, m2, m3;
float avg;
char grade;
};

2. Type definition structure.
Syntax:-

typedef struct
{
datatype member1;
datatype member2;
:
:
} Type name;

→ where typedef is a keyword added to the beginning of defmition.
→ Struct is the keyword which tells the complier that a structure is being defined.
→ member1, member2, .. membern are called members of the structure.
→ The members are declared with in the curuly braces.
→ The closing bracee must end with type defmition name which mturn ends with semicolon (;)

Ex:-
```
typedef struct
{
    char sname[20];
    char rno[15];
    int m1, m2, m3;
    float avg;
    char grade;
} student;
```

Note:- All variables (members) of the structure are declared with in the structure.
→ During the declaration of structure the memory is not allocated to the structure.

Structure variable declaration:-

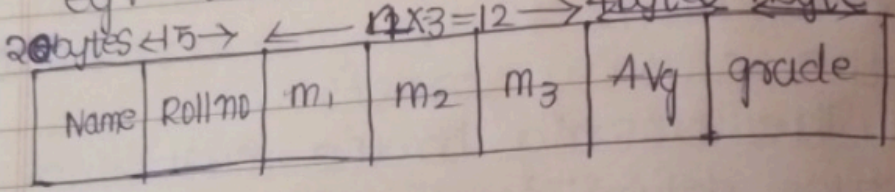
→ when we declare a variable to the

structure then only the memory will be allocated to the structure.

Tagged Structure :-

Syntax :- struct tagname var1, var2;

Eg :- struct student s1, s2;



$S_1 = 20+15+12+4+1$

$S_1 = 52$ bytes.

Type defmition Structure :-

type name var1, var2, ... ;

student s1, s2;

same as tagged = 2 bytes.

Initialization :-

```
struct student
{
    char name[20];
    char rNO[15];
    int m1, m2, m3;
    float avg;
    char grade;
};
```

s={'xxx',"22R1A0549",25,26,27,28,'A'};

Accessing the memory of Structure:-

2 operators.

1. Dot operator: Structure variable is a normal variable
2. Arrow operator: Structure variable is a pointer variable.

Normal variable → Structure variable → Structure variable.

pointer variable → Structure variables → Structure number.

Ex: struct sample
{ int x;
float y;
char c;
};
struct sample s, *s1;
s.x (s1→x)
s.y (s1→y)
s.c (s1→c)

(10) write a c program to read and print Student details as Student name.

#include<stdio.h>
Struct student

```
{
  char sname[20];
  char rno[15];
  int m1, m2, m3, total;
  float avg;
  char grade;
} s;
int main()
{
printf("enter student name, roll no,
      3 subject marks);
scanf("%s %s %d %d %d", s.sname,
s.r.no, &s.m1, &sm2, &sm3);
s.total = s.m1 + s.m2 + s.m3;
s.avg = s.total/3.0;
if (avg >= 75)
   s.grade = 'A';
else
   s.grade = 'B';
printf("Student name = %s\n", s.name);
printf("Roll no = %s\n", s.Rno);
printf("Total = %d\n", s.total);
printf("Average = %f\n", s.avg);
printf("Grade = %c\n", s.grade);
}
```

Q. Write C program to read employee details as employee name, employee Id, employee basic salary and

print name, Id, and gross salary.
BS, HRA, DA, PF.
S = employee.details.

```
# include <Stdio.h>
struct employee
{
   char ename[20];
   char eId[20];
(X)  int BC, HRA, DA, PF, gross salaray;
} S;
int main()
{
printf("enter employee name, Id, Basic
   Salary, HRA, DA, PF");
Scanf("%s%s%d%d%d", ename,
 eId, & eB.C, & eHRA, & ePF, & eDA);
e.gross salary = BC+DA+HRA-PF;
printf("employee name = %s\n", ename
printf("employee Id = %s\n", eId);
printf("gross salary = %d\n", egross
                salary);
}

# include < stdio.h >
struct employee
{
   char ename[20];
   char eId[20];
```

```
float DA, HRA, gross salary;
int PF, BC;
} e;
int main()
{
  printf("enter employee name, Id, BC, PF");
  Scanf("%s%s%d%d", e.name, e.Id,
        &e.BC &e.PF);
  DA = 0.1 * BC;
  HRA = 5/100 * BC;

  gross salary = BC+DA+HRA-PF;
  printf("employee name = %s\n",
    employee Id = %s\n, gross salary = %f\n;
  e.name, e.Id, e.gross salary);
}
```

Nested structure :-

→ One structure present inside the another structure is called as nested structure.

→ Any number of structure can be nested but not more than three.

Syntax :-

```
struct Tag-name1
{

};
```

```
float DA, HRA, gross salary;
int PF, BC;
} E;
int main()
{
 printf("enter employee name, Id, BC, PF");
 Scanf("%s%s%d%d", e.name, e.Id,
      &e.BC &e.PF);
 DA = 0.1 * BC;
 HRA = 5/100 * BC;

 gross salary = BC+DA+HRA-PF;
 printf("employee name = %s\n",
  employee Id = %s\n, gross salary = %f\n",
 e.name, e.Id, e.gross salary);
}
```

Nested structure:-

→ one structure present inside the another structure is called as nested structure.

→ Any number of structure can be nested but not more than three.

Syntax:-

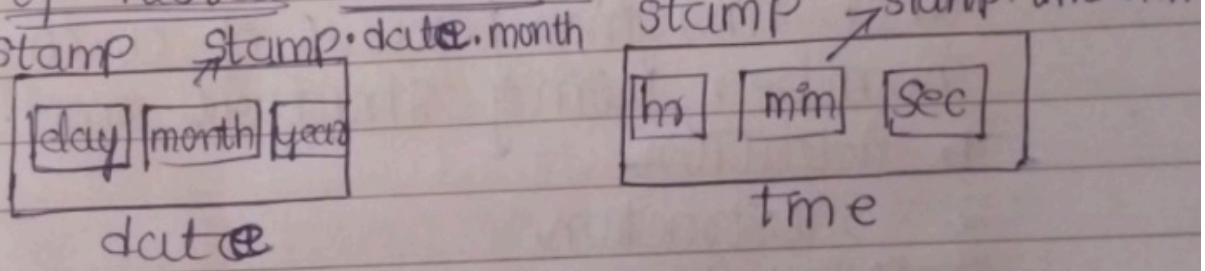
```
Struct Tag-name1
{

};
```

```
struct tag-name2
{

};
struct tag-name3
{
  Struct Tag-name1;
  Struct Tag-name2;
};
Struct Tag-name 3 variable;
```

Declaration, initialization and accessing of Nested Structures:-




```
# include<stdio.h>
struct date
{
   int day, month, year;
};
struct time
{
  int hr, min, sec;
};
struct stamp
{
```

```
    struct date d;
    struct time t;
} s;
int main()
{
    struct stamps s = {{7, 11, 2023}, {10, 25, 30}};
printf("Date = %d-%d-%d", s.d.day,
        s.d.month, s.d.year);
printf("time = %d:%d:%d", s.t.hr,
        s.t.min, s.t.sec);
}
```

(Q) write a c program to read real and imaginary values of a complex number using structure and find
1. Addition
2. Substraction
3. Multiplication.

```
struct complex
{
    int real, img;
} c1, c2, c3;
int main()
{
printf("enter complex number 1:");
scanf("%d%d", &c1.real, &c1.img);
printf("enter complex number 2:");
scanf("%d%d", &c2.real, &c2.img);
```

```
c3.real = c1.real + c2.real;
c3.img = c1.img + c2.img;
printf("Addition of 2 complex numbers=
%d+i %d\n", c3.real, c3.img);
printf("Substraction of 2 complex numbers=
%d-i%d\n", c3.real, c3.img);

c3.real = (c1.real * c2.real) - (c1.img *
          c2.img);
c3.img = (c1.real * c2.img) + (c1.img * c2.real);
```

Array of structures :-

If a structure variable is declared as an Array then it is known as array of structures

Syntax for Array of structure declaration.

```
struct tag_name array of structure [size];
                (or)
Structure variables.
```

Ex:- Let us take an example to store the information of 3 students we can have the following structure defination and declaration,

```
Struct Student
{
  char name[10];
   int rno;
   float avg;
};
Struct Student S[3];
```

or

```
} S[3];
```

Initialization of Array of Structures:-

```
Struct Student S[3] = {{"ABC", 1, 56.7},
{"xyz", 2, 65.8}, {"pqr", 3, 82.4}};
```

(9) write a c program to store name, roll number, year and marks of three subjects of n students and print the student name roll number grade average, and grade based on the marks of average of the student using structures.

```
#include <stdio.h>
Struct Student
{
   char name[100];
   char rno[100];
    int y, m1, m2, m3;
} S1[100];
int main()
{  int n, i;
   float avg;
   char g;
printf("Enter n value:");
Scanf("%d", &n);
printf(" Enter roll no, name, year, m1, m2,
                    m3 ;");
```

```
for(i=0; i<n; i++)
scanf("%s%s%d%d%d%d%d", s1[i].rno,
s1[i].name, &s1[i].y, &s1[i].m1, &s1[i].m2,
&s1[i].m3);
for(i=0; i<n; i++)
{ avg=(s1[i].m1+s1[i].m2+s1[i].m3)/3;
  if(avg>=75)
    g='A';
  else if(avg>=50)
    g='B';
  else
    g='C';
printf("\n Roll no=%s\n name=%s\n Average
=%f\n Grade=%c\n", s1[i].rno, s1[i].name,
avg, g);
}
};
```

8. Write a c programe to read employee details as employee no, elpoyname, Basis salary, HRA, DA of n employee using structures and print employee no, employee name and gross salary of n employee.

```
#include<stdio.h>
struct employee
{   char name[100];
    char no[100];
    int BS, HRA, DA;
} e[100];
int main()
```

```
{ int n, i;
int Gross salary;
printf("enter n value");
Scanf("%d", &n);
printf("enter name, no, BS, HRA, DA");
for(i=0; i<n; i++)
scanf("%s%s%d%d%d", &E[i].name,
E[i].no, &E[i].BS, &E[i].HRA, &E[i].DA);
for(i=0; i<n; i++)
{ Gross salary = BS+HRA+DA;
printf("\n Eno=%os, E.name=%os,
Gross salary = %d",
E[i].no; E.[i].name,
Gross salary);
gross salary= e[i].bs+ e[i].hra+e[i].da;
```

Structures and functions :-

→ Structures are more useful if we are able to pass them to the functions and return them.

→ The structures members can be passed to the function in various ways as shown below.

Call by value
1. By passing individual members of structure
2. By passing the whole structure

Call by reference
3. By passing structure through pointers

passing individual members of structure:

actual parameters - members of the structure
formal parameters - normal variables like int, float.

Q. write a c program to read numerator and denominator for a fraction using structures and find multiplication of two fractions by passing individual members of structure.

```
# include<stdio.h>
int multiply(int x, int y);
struct fraction
{
  int n,d;
} f1,f2,f3;
void main()
{ printf("enter the fraction\n");
scanf("%d%d",&f1.n,&f1.d);
printf("enter the 2 fraction 2\n");
scanf("%d%d",&f2.n,&f2.d);
f3.n=multiply(f1.n,f2.n);
f3.d=multiply(f1.d,f2.d);
printf("result=%d%d\n",f3.n,f3.d);
}
int multiply(int x, int y)
{
 return(x*y);
}
```

2. PASS THE WHOLE STRUCTURE:

→ Actual Arguments are the names of the structure variables.
→ The formal Arguments are structure variables with specification of structure type.

```
#include<stdio.h>
Struct fraction multiply(Struct fraction f1,
        Struct fraction f2);
Struct fraction
{
 int n,d;
} f1,f2,f3;
void main()
{ printf("Enter the fraction\n");
 Scanf("%d%d",&f1.n,&f1.d);
printf("Enter the 2 fraction\n");
 Scanf("%d%d",&f2.n,&f2.d);
f3 = multiply (f1,f2);
printf("Result = %d/%d \n", f3.n,f3.d);
}
Struct fraction multiply (Struct fraction f1,
  Struct fraction f2)
{
 Struct fraction res;
 res.n = f1.n * f2.n
 res.d = f1.d * f2.d;
  return res;
}
```

3. passing structure through pointers:-

→ Actual parameters → & structure variable
Address of structure variables

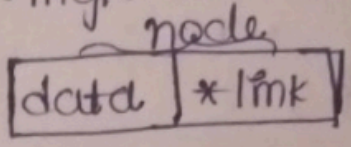
→ Formal parameters → structure variables are declared as pointer variables.

```
#include<stdio.h>
struct fraction multiply(struct fraction *f1,
  struct fraction * f2);
{
  int n,d;
}
void main()
{
  struct fraction *f1,*f2,*f3,a,b,c;
  f1 = &a;
  f2 = &b;
  f3 = &c;
printf("enter the fraction numerator\n");
Scanf("%d", &f1->n);
printf("enter the fraction denomenator\n");
scanf("%d", &f1->d);
printf("enter the 2nd fraction numerator\n");
scanf("%d", &f2->n);
printf("enter the 2nd fraction denomenator");
Scanf("%d", &f2->d);
*f3 = multiply(f1,f2);
printf("result = %d/%d\n", f3->n, f3->d);
}
```

```
struct fraction multiply (struct fraction
* f1, struct fraction * f2)
{
  struct fraction * res, d;
  res = &d;
  res->n = f1->n * f2->n;
  res->d = f1->d * f2->d;
  return * res;
}
```

Self referential functions :- Structures

→ A structure which refers to itself is known as self referential structure.
→ It is mainly used in implementation of data structures.
→ Example : to create a single linked list.

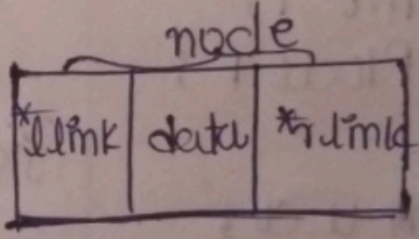



```
struct node
{ int data;
  struct node *link; → self-referential
}
```

→ In the above example node is pointing to node again until link assigned with null.

example :- to create a node in double linked list

```
struct node
{ struct node *llink;
  int data;
  struct node *rlink;
}
```



In the above example node is pointing to node again until link assigned with null.

UNION :-

→ Union is a derived datatype.
→ Union is a collection of dissimilar dataitems stored in continuous memory locations under a single name.

Note :- The syntax declaration and use of Union is similar to the structure but

The functionality is different.

Syntax:-

```
union tag-name
{ type1 member;
  type2 member;
  ....
};
```

Variable declaration is similary like structure.

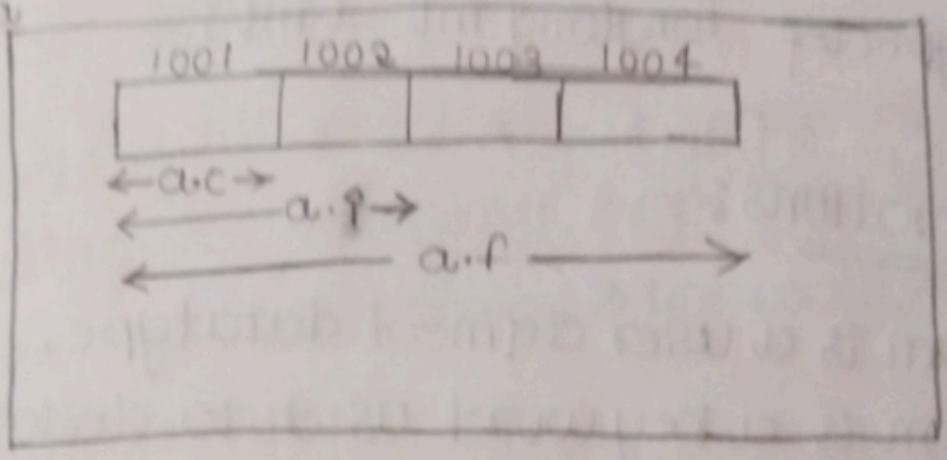
Syntax:- union tag-name var1, var2...;

union u { char c; int i; float f; }; union u a; tagged union	union u { char c; int i; float f; }a; variable union	typedef union { char c; int i; float f; }U; Ua; typeclef union.

→ we can access various members of the union as mentioned below.

a.c → 1 byte
a.i → 2 byte
a.f → 4 byte

for string (s) → 20 byte.




```
#include <stdio.h>
union student
{ char name[20];
  int marks;
}u;
void main()
{ printf("enter name\n");
  scanf("%s", u.name);
  printf("name=%s", u.name);
  printf("enter marks\n");
  scanf("%d", &u.marks);
  printf("Marks=%d\n", u.marks);
}
```

Typedef :-

→ Type def is used to redefine the name of an existing variable type

Syntax :- typedef data-type identifier;

(or)

typedef existing datatype newdatatype

Ex:- typedef longlong int dint;

Enumerations:-

→ Enum is a user defined datatype.
→ Enum is a keyword used to declare new enumeration types.
→ It is mainly used to assign names to the integral constants (sequence of digits without a decimal point).

Syntax:- enum identifier{const1, const2, ...};

enum - is the keyword
identifier - name of enumeration.

By default, the values of the constants are as follows :-

const1 = 0, const2 = 0

```
#include<stdio.h>
enum weekday{sun, mon, tue, wed, thr,
             fri, sat};
             0   1   2   3   4   5   6
void main()
{
  enum weekday today;
  today = sat;
  printf("today = %d\n", today);
}
```

Output :- 6