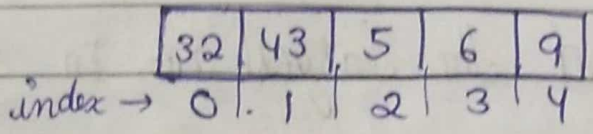
Unit-4 ARRAY

Date : __/__/__
Page : _____

Array - An array is a collection of similar data types stored in a contiguous memory location.
(continue (one by one))

Representation - Data type name [5]
↳ size of array (compulsory)

Arr



[Packed]

Array is a 0 base index

⇒ In an array of size n, the index ranges from 0 to (n-1)

For pre-defined variable - int arr[3] = {30, 2, 5}

For input from user - int arr[3];
scanf ("%d", &arr[2]);
It shows the second index

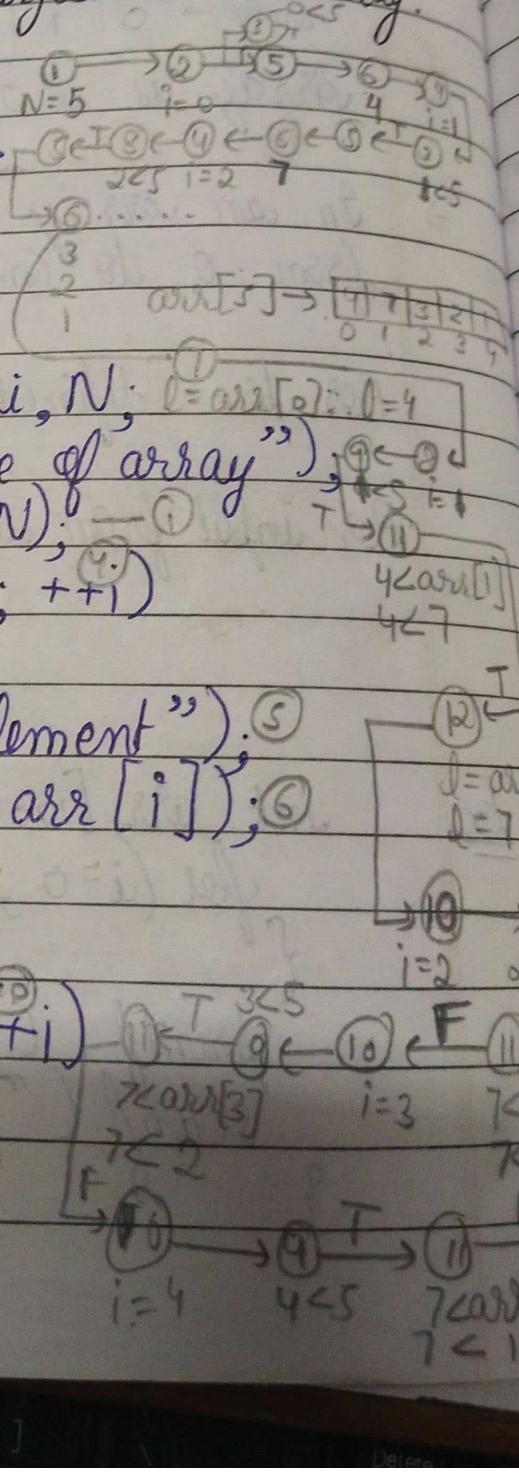
Ex
```
int arr[100], i;          // n-1 = 100-1 = 99.
for (i=0; i<=99; i++)
{
printf ("Enter elements");
scanf ("%d", &arr[i]);
}
```

Ex

```
int arr[100], i, N;
printf ("Enter the term");
scanf ("%d", &N);
for( i=0; i< N; i++)
{
  printf ("Enter elements");
  scanf ("%d", & arr[i]);
}
```

Q WAP to find the largest no. in array.

```
#include <stdio.h>
#include <conio.h>
int main()
{
clrscr();
int arr[50], l, i, N;
printf ("Enter size of array");
scanf ("%d", &N);
for (i=0; i< N; ++i)
  {
    printf ("Enter element");
    scanf ("%d", & arr[i]);
  }
l= arr[0];
for (i=1; i<N; ++i)
  {
  if (l< arr[i])
```



Date __/__/__
Page

```
{ d = arr[i]; }
}
printf("largest element = %d", d);
getch();
return 0;
}
```

Linear Search

a[5] → N = 5

5	9	7	4	10
0	1	2	3	4

index →

search is dn 1 by 1 from 0th index to last

For eg we have to search e = 4

```
for(i=0; i<N; ++i)
{
  if(a[i] == e)
  {
    printf("%d is present at loc = %d", e, i);
    break;
  }
}
if(i == N) { printf("%d not found", e); }
```

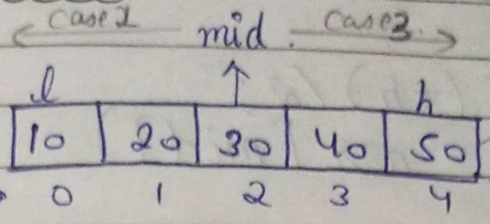
1 → 2 (0<5, T) → 4 (a[0]=5, 5==e, 5==4) F → 3 (i=0+1, i=1) → 2 (1<5, T) → 4 (a[1]=9, 9==4) F → 3 (i=1+1, i=2) → 4 (a[2]=7, 7==4) F → 3 (i=2+1, i=3) → 4 (a[3]=4, 4==4) T → 5 (4 is present at loc = 3) → 6 (out of loop, i==5) → 7 → 9 (5<5, F) → 13 (largest element = 7)

Q WAP to search a no. entered by user and its location.

```
#include <stdio.h>
#include <conio.h>
{
int main ()
int arr[30], N, e, i;
printf ("Enter no. of elements:");
scanf ("%d", &N);
for (i=0 ; i<N; ++i)
  {
  printf ("Enter element : " );
  scanf ("%d", &arr[i]);
}
printf ("Enter element that is to be searched
scanf ("%d", e);
for(i=0; i< N; ++i)
  {
    if (arr[i] == e)
    {
    printf ("%d is at loc = %d", e, i);
     break;
    }
}
if (i== N)
{
printf ("%d is not found", N);
}
getch();
return 0;
}
```

Date : __/__/__
Page: _____

Binary Search



search e = 40

l = 0 h = n-1 = 5-1 = 4

mid = (l+h)/2 = 2

We first compare e with mid then

case 1: if it is equal to mid then print that

case 2: if it is "less than" then search on LHS of mid

case 3: if it is greater than " " " " RHS of mid.

```
#include<stdio.h>
int main()
{
 int arr[30], N, e, i, l, h, mid;
 printf("Enter no. of elements");
 scanf("%d",&N);                    Let N = 4
 for(i=0; i<N; ++i)
 {
  printf("Enter element of array:");
  scanf("%d", &arr[i]);
 }
```

arr[4] →

5	6	7	9
0	1	2	3

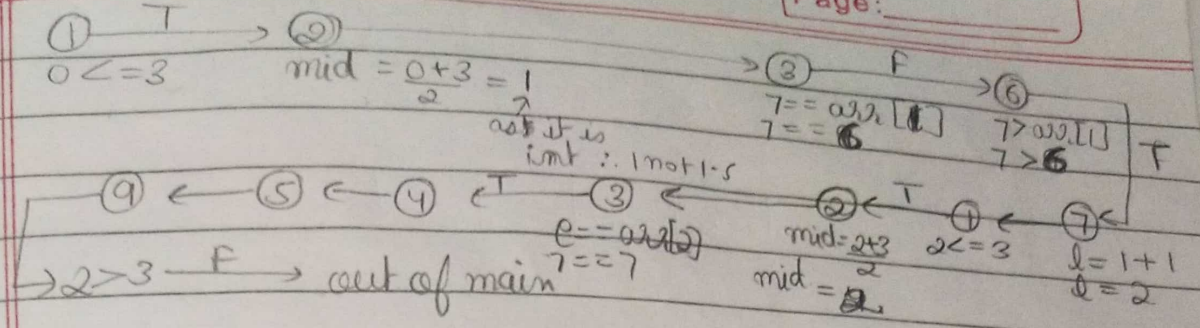
```
 l = 0                    l = 0
 h = N-1;                 h = n-1 = 4-1 = 3
 printf("Enter element to be searched");
 scanf("%d",e);
```

(i) Let e = 7 (ii) Let e = 5 (iii) Let e = 2

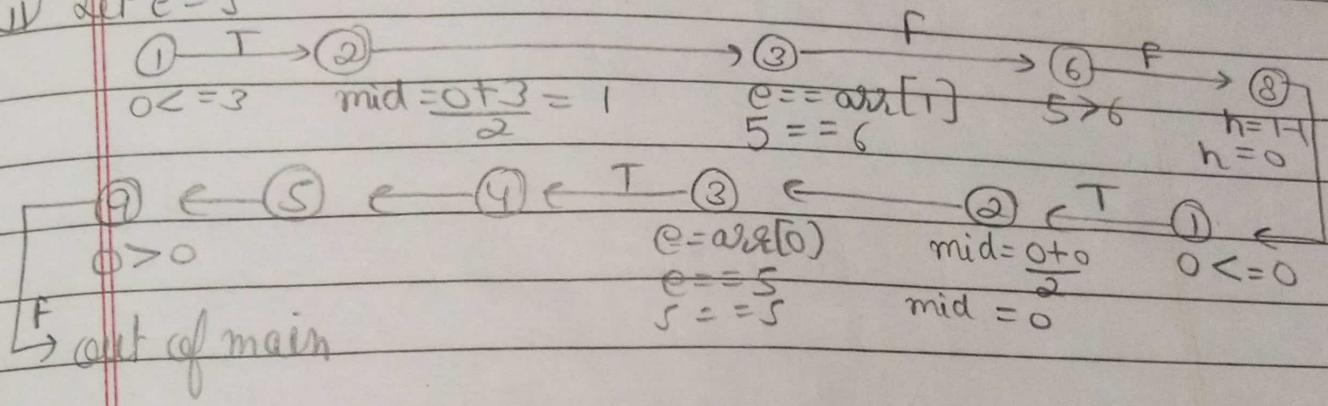
```
while (l<=h) ①
{
   mid=(l+h)/2; ②
  if (e==arr[mid]) ③
  {
  printf("%d is found at loc = %d", e, mid+1); ④
    break; ⑤
  }
  else if (e>arr[mid]) ⑥
    {
       l= mid+1; ⑦
    }
else
    {
      h= mid-1; ⑧
    }
}
  if (l>h) ⑨
  {
   printf ("%d is not found", e); ⑩
}
}
return 0;
}
```

Date : __/__/__
Page: _____

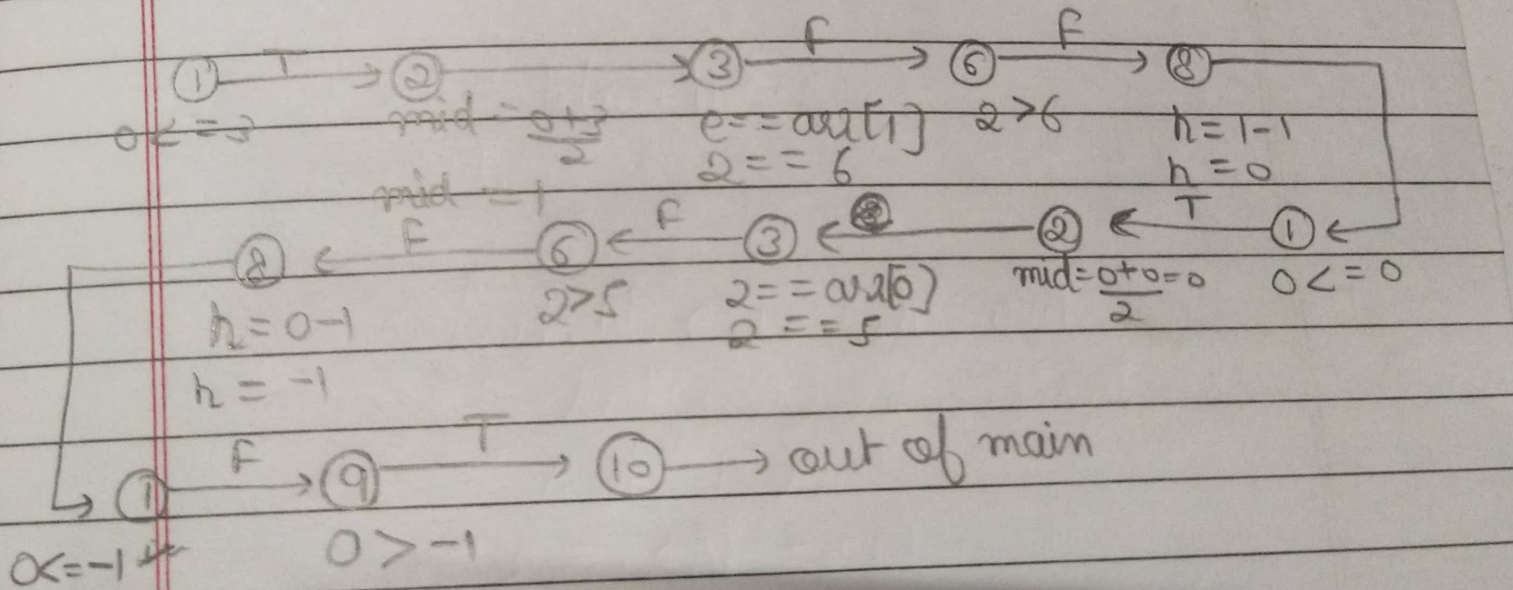
(i)



(ii) Let e = 5



(iii) Let e = 2



2D Array

Syntax: Data type Name [Row] [column]
↳ size of column
↳ size of Row

- ☆ int arr [4] [4] ← for sq. matrix
- ☆ int arr [2] [3] ← 2 Row 3 columns matrix

⇒ 2D array can be defined as an array of array

Eg

int a[2][2]

	Column1	Column2
Row1	a[0][0]	a[0][1]
Row2	a[1][0]	a[1][1]

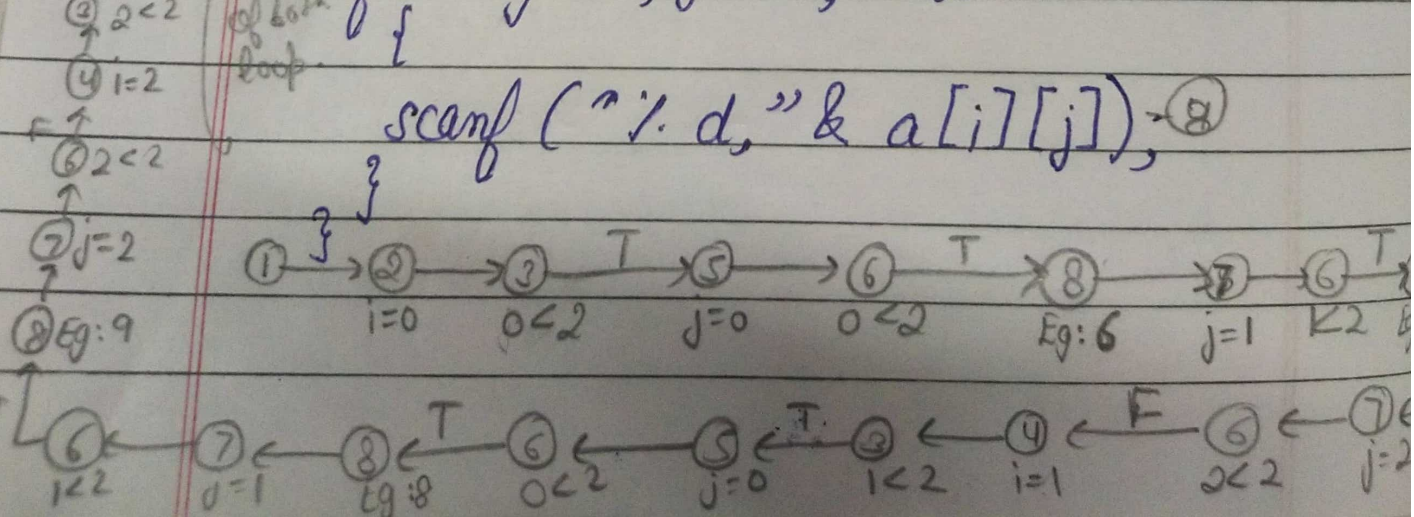
← This is how indexing is done.

⇒ Let N=2 ← no. of Row = no. of column for sq. matrix

```
printf ("Enter the element of array"); -①
for (i=0; i<N; ++i)
{
    for (j=0; j<N; ++j)
    {
        scanf ("%d," & a[i][j]); -⑧
    }
}
```

i=Row
j=Column

How to take input.



Date : __/__/__
Page : _____

Q Addition of 2 matrices A & B

C = A + B.

```
int A[100][100], i, j, N, C[100][100], B[100][100];
Let N=2;
printf("Enter the elements of array A");
for( i=0; i<N; ++i)
  {
  for( j=0; j<N; ++j)
    {
      scanf("%d", &A[i][j]);
  }}printf("Enter elements of array B");
for( i=0; i<N; ++i)
{
for( j=0; j<N; ++j)
  {
    scanf("%d", &B[i][j]);
  }
}
for( i=0; i<N; ++i)
  {
  for( j=0; j<N; ++j)
  {
  C[i][j] = A[i][j] + B[i][j];
  }
}
printf("Sum = %d", C[i][j]);
for( i=0; i<N; ++i)
{
  for( j=0; j<N; ++j)
  {printf("%d \t", C[i][j]);
  printf("\n");}}
```

C = [a[0][0] a[0][1]; a[1][0] a[1][1]] + [b[0][0] b[0][1]; b[1][0] b[1][1]]

\t = tab = space.

Date : __/__/__
Page: _____

Q WAP to print the transpose of matrix

```
#include <stdio.h>
#include <conio.h>
void main()
{
int i, j, N, A[100][100], B[100][100];
printf("Enter no. of rows & colums:");
scanf("%d", &N);
printf("Enter elements of array matrix:");
for(i=0; i<N; ++i)
{
    for(j=0; j<N; ++j)
    {
    scanf("%d", &A[i][j]);
    }
}
for(i=0; i<N; ++i)
{
 for(j=0; j<N; ++j)
 {
   B[i][j] = A[j][i]   <- main formula for Transpose
 }
}
for(i=0; i<N; ++i)
{
 for(j=0; j<N; ++j)
 {
```

```
printf("%d \t", B[i][j]);
}
printf("\n"); }
getch();
}
```

8 WAP to print rectangular matrix

```
#include <stdio.h>
void main()
{
int i, j, R, C, A[100][100];
printf("Enter no. of rows R:");
scanf("%d", &R);
printf("Enter no. of columns C:");
scanf("%d", &C);
for(i=0; i<R; ++i)
{
  for(j=0; j<C; j++)
  {
  printf("%d \t", A[i][j]);
  }
  printf("\n");
}
}
```

Rows → ← Columns

Date : __/__/__
Page : _____

Matrix Multiplication

$$A\begin{bmatrix} A[0][0] & A[0][1] \\ A[1][0] & A[1][1] \end{bmatrix} \times B\begin{bmatrix} B[0][0] & B[0][1] \\ B[1][0] & B[1][1] \end{bmatrix}$$

$$= C\begin{bmatrix} C[0][0] & C[0][1] \\ C[1][0] & C[1][1] \end{bmatrix}$$

$C[0][0] = A[0][0] \times B[0][0] + A[0][1] \times B[1][0]$

$C[0][1] = A[0][0] \times B[0][1] + A[0][1] \times B[1][1]$

$C[i][j] = C[i][j] + A[i][k] \times B[k][j]$

↳ done like `int sum = 0`, `sum = sum + a`

```
for(i=0; i<N; ++i)
{
  for(j=0; j<N; ++j)
  {
    for(k=0; k<N; ++k)
    {
      C[i][j] = C[i][j] + A[i][k] * B[k][j];
    }
  }
}
```

Date : __/__/__
Page : _____

Q WAP to multiply the matrix :-

```
#include <stdio.h>
#include <conio.h>
int main()
{
int i, j, k, N, A[100][100], B[100][100], C[100][100]={0};
printf("Enter no. of Rows & Columns: ");
scanf("%d",&N);
printf("Enter elements of matrix A: ");
for(i=0; i<N; ++i)
{
  for(j=0; j<N; ++j)
  {
    scanf("%d",& A[i][j]);
  }
}
printf("Enter elements of matrix B: ");
for(i=0; i<N; ++i)
{
  for(j=0; j<N; ++j)
  {
    scanf("%d",& B[i][j]);
  }
}
for(i=0; i<N; ++i)
{
  for(j=0; j<N; ++j)
```

```
    {
    for (k=0; k<N; ++k)
    {
  C[i][j] = C[i][j] + A[i][k] x B[k][j];
    }
  }
}
 for (i=0; i<N; ++i)
  {
   for (j=0; j<N; ++j)
   {
     printf ("%d \t", C[i][j]);
   }
   printf ("\n");
}
 getch();
 return 0;
}
```

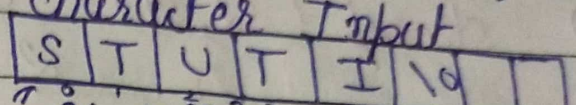
Date : __/__/__
Page : _____

STRING

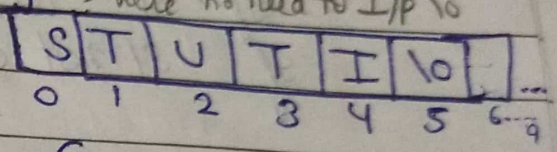
→ A string is nothing but a null (\0) terminated character array.

→ Data Type String name [ ] ← size

→ [char str [10] = { 'S', 'T', 'U', 'T', 'I', '\0' }
↳ Character by character Input



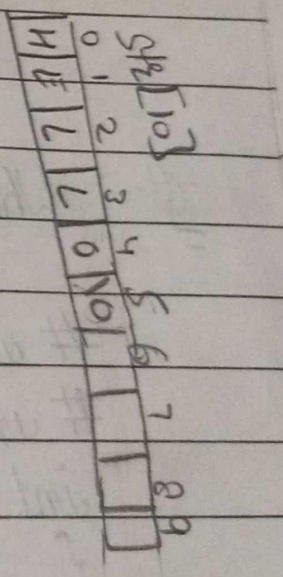
→ [char str[5] = { "Stuti" } ← here no need to I/P \0
↳ Input as a whole



→ Example of taking I/P from user (character by character)

```
char str [10];
for (i=0; i<5; ++i)
{
  scanf ("%c", &str[i]);
}
str [i] = '\0';
```

when i=5



→
```
char str [10];
printf ("Enter string");
gets (str); ← Take I/P as a whole in str
puts (str); ← Print str
```

Functions of string (s)

(s) ← name of string

① strlen(s) – find length of string (\0 is not counted)

② strcpy(s2, s1) – To copy element of s1 to s2
s1 का content s2 में store

③ strcat(s2, s1) – joins s1 into s2

④ strcmp(s2, s1) – to compare characters of 2 strings
~~each character~~ should be same exactly
If s1 == s2 → returns 0
If s1 not equal to s2 → return any ASCII value except 0

⑤ strrev(s) – to reverse string
Example s = AMAN
strrev(s) = NAMA

Q WAP to check wether the string is Pallindrome or not

```
#include<stdio.h>
#include<string.h>
int main()
{
char s1[10], s2[10];
printf("Enter a string:");
gets(s1);
strcpy(s2, s1);
strrev(s2);
```

```
if( strcmp(s2, s1) == 0)
{
  printf("%c is Pallindrome", s1);
}
else
{
  printf("%c is not Pallindrome", s1);
}
return 0;
}
```

Structure

Structure - A collection of values of similar or different data types. A structure is a users defined data type.

→ A keyword 'struct' is used to define structure

→ **Syntax**

```
struct struct_name
{
datatype variable;
"       "
"       "
};
```

→ For Student Data

Example

```
struct student
{
 char name[50];
 int roll_no;
 float marks;
}s1, s2;

int main()
{
  s1.roll_no = 4032;
  s1.Name = "XYZ"; s1.marks = 98.72;
  printf("%c %d %f", s1.Name, s1.rollno, s1.marks);
    s2.roll_no = 4033;
    s2.Name = "ABC";
    s2.marks = 90.93;
printf("%c %d %f", s2.Name, s2.roll_no, s2.marks);
}
```

for 2 students s1 & s2

Example

```
int main()
{
 if (s1.marks > s2.marks)
 {
  printf("%c marks is greater than %c", s1.marks,
                                        s2.marks);
 }
```

else
{
printf
}

Example To

index of array a[i]

```
else
{
printf ("%c is greater marks than %c", s2.marks,
                                        s1.marks);
}
```

Example To take input from users

```
struct student
{
char Name[50];
int roll_no;
float marks;
} a[i];
```

in array format
a[0] will have student 1 details
a[1] " " " 2 "
& so on.

```
int main()
{ int i;
printf ("Enter student details");
for(i=0 ; i<50; ++i)
{
scanf ("%c %d %f", &a[i].Name, &a[i].roll_no,
                   &a[i].marks); }
for (i=0; i<50; ++i)
{
if (a[i].marks < 60).
{
printf ("%c", a[i].Name);
}
}
return 0; }
```

index of array a[i]

to take input for details of 50 students

for printing name of students who got marks < 60

Date : __/__/__
Page : _____

4/06/2023

Union

Def → It is similar to structure and is a collection of values of different data types. It is also user defined data type.

→ **Keyword for union** → 'union'

→ **Syntax**

```
union tag_name
{
data_type variable1;
"  "  "  2;
"  "  "  3;
"  "  "  n;
};
```

जो सबसे large memory वाला data type होगा तो Backend में उस data type की memory create होगी बस बाकी की नहीं होती

→ **Example**

```
union A
{
int a;
float b;
};
```

2 bytes, 4 bytes

only memory of float is created in backend.

Example

```
union u1
{
int a;        2 byte
float b;      4 byte
char s[10];   1 byte
}
```

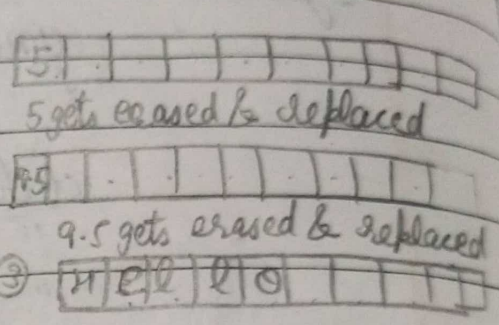
but size is 10 so size of structures 10 bytes
∴ largest size data type available is char so its memory is created

```
void main()
{
union u1 z;
z.a = 5;  -①
z.b = 9.5;  -②
z.s = "Hello";  -③
printf("%c", z.s);
```

user defined data type

data type of z

→ O/P → Hello



मतलब अब यहां z.a या z.b या z.c कुछ भी print करवाने पे Hello ही o/p आएगा क्योंकि वो ही last में store हुआ था।

⇒ If we want to print every value then do this

```
z.a = 5;
printf("%d", z.a);
z.b = 9.5;
printf("%f", z.b);
z.c = "Hello";
printf("%c", z.c);
```

Date : __/__/__
Page : _____

Properties of Union:

* Union can be defined with many members but 1 member can access a value at a time.
* Efficient way of using the same memory location for multiple purposes.

Difference b/w Structure & Union

Structure	Union
* Keyword 'struct' is used to define structure	Keyword 'union' is used to define union.
* Every member within structure is assigned a unique memory location.	Largest data type's memory location is shared by all data members.
* Changing the value of one data member will not affect other data member in structure	Changing value of 1 data member will affect other data member in union.
* The total size depends on the sum of the size of every data member.	The total size depend on the size at the largest data member

Date : __/__/__
Page :

* We can access any member at a time	Only 1 member can be accesed at a time

* Syntax
struct structname
{
data_type variable1;
⋮
datatype variablen;
};
→ K
	Syntax
	union tagname
	{
	data_type member1
	⋮
	data_type membern
	};

typedef

definition of typedef.

Def → It allows user to provide alternate name for primary and user defined data types.

For eg we want to write Ind instead of int

int A ⟶ Ind A

for this we do ↓

```
typedef int Ind;
```

→ Example for use of typedef in user defined data type

```
struct student
{
  int roll;
  char name[10];
};
```

```
struct student s1;
type struct student AK;   (this can be written as)   (now)
   AK s1;
```

Date : __/__/__
Page : _____

Enumeration - It is used to assign names/identifiers to the integer constant. Each integer values assign an identifier.

→ **Keyword** 'enum'

→ **Syntax:** enum enum_name {identi1, identi2, ..., identin}; (identifier)

→ **Example:** enum colors {Red, Blue, Black, Green, Yellow};
index → 0 1 2 3 4

User defined data type

```
#include <stdio.h>
void main()
{
 enum colors{Red, Blue, Black, Green, Yellow}
 enum colors Y;
   Y= Black;
 printf("%d", Y); ⟶ Output = 2
```

If I would have written
Red = 5; (indexing from 5)
then Output = 7

Red	Blue	Black	Green	Yellow
5	6	7	8	9

OR

Black = 15
Y = Green
O/P = 16

Red	Blue	Black	Green	Yellow
0	1	15	16	17