UNIT-II Control Structures

→ i) conditional statement
→ ii) Decision making statement
→ iii) Selection statement

Iterate / Repetitive / looping

i) for loop
ii) while loop
iii) do while loop

i) Simple if
ii) if else
iii) Nested if else
iv) else if ladder
v) Switch case

O.S → other statements

break
continue

Syntax:

```
if(condition)
{
  // True statement
}
// other statement.
```

Ex:

```
if(n>10)
{
printf("value is >10");
}
printf("out of simple
```

⇒ if else:

```
if(condition)
{
   // True statement
}
else
 {
  // false statement
 }
// other statement.
```

Ex:

```
IF(n>10)
{
pf("value is greater than 10");
}
 else
 {
 pf("value is less than 10");
 }
 pf("out of if else");
```

---

ii) ~~Nested if else~~ ✗

```
if(condition)
{
  // True statement
}
if else(condition)
{
  // True statement
}
else
  {
   // false statement
  }
// other statement.
```

ii) Nested if else: ex: if (n>10)

```
if (condition1)
{
    if (condition2)
    {
        // Statement1
    }
    else
    {
        // Statement2
    }
}
else
{
    // Statement3
}
// other Statement.
```

```
{
    if (n<20)
    {
    pf ("values is b/w 10 to 20");
    }
    else
    {
    pf ("value is gt 20");
    }
}
else
{
    pf ("value is less than 10");
}
pf ("out of nestedif else");
```

iv) if else ladder :

```
if (condition)
{
  // statement1
}
else if (condition2)
{
  // statement2
}
else
  :
  :
  if (condition n)
  {
    // statementn
  }
  else
  {
    // default statement
  }
// other statement.
```

Ex:

```
if (n>50)
{
  pf("The value is greater than 50");
}
else if (n>40)
{
  pf("The value is g.t 40");
}
  else if (n>30)
  {
    pf("value is g.t 30");
  }
  else
  {
    pf("value is l.s 30");
  }
printf("out of else if ladder");
```

⇒ Switch

```
Switch (option)
{
case opt-val1 : statement1 ;
                break;
case opt-val2 : statement2 ;
                break;
case opt-val3 : statement3 ;
                break;
default : default statement;
}
// other statements.
```

Eg:

```
int main( )
{
  char ch;
  pf("enter your choice:");
  Scanf("%c", &ch);
  Switch (ch)
  {
  case 'R': printf("Red");
            break;
  case 'O': pf("orange");
            break;
  default : pf("Invalid choice
  }
  pf("out of switch case");
}
```

⇒ For loop : counter controlled loop

⇒ while loop : event controlled loop } pre-test loops

⇒ do while loop : event controlled loop - post-test loop

⇒ for loop:

```
for(initialization; condition; updation)
{
   // body of the loop.
}
   // other statement.
```

Note: If the no. of iteration are fixed we should use for loop.

Ex: write a c-program to print 1st 20 natural no

```
#include <stdio.h>
int main()
{
 int i;
 for(int i=1; i<=20; i++):
 {
    printf("%d\n", i);
 }
return 0;
```

"%d" — 1234...

"%d" — 1 2 3 4

"\n%d" (or) "%d\n" — 1 2 3 4

"\t%d" (or) "%d\t" — 1 2

=> while loop:

```
while (condition)
{
    // body of the loop
}
// other statement.
```

Ex: write a c program to find the sum of n integers.

Ans
```
#include<stdio.h>
int main()
{
int num, sum=0;
while (n
```

Ans
```
#include <stdio.h>
int main()
{
int num,sum=0;
```
pf("enter the numbers (0-to exit):");
```
scanf("%d", &num);
while (num!=0)
{
num=sum+num;
scanf("%d" &num);
}
printf("sum=%d", sum);
}
```

pf("enter the no's (0 to e
```
while (num==0)
{
scanf("%d", &num
sum=sum+num;
```

⇒ do while

Syntax:

```
do
{
  //body
} while(condition);
```

Ex: sum of n integers.

```
{
int num, sum = 0;
pf("Enter the numbers
    (0 - to exit):");
do
{
  scanf("%d", &num);
  sum = sum + num;
} while(num != 0);
printf("sum = %d", sum);
}
```

Difference b/w while loop & do while loop.

while loop	do while loop
1) pre test	post test
2) Event controlled	event controlled
3) minimum = 0	min = 1
4) syntax:	Syntax:

```
while (condition)
{
  //body of the loop
}
```

```
do
{
  //body of the lo
} while (cond
```

Break and Continue statements:

⇒ These statements are ~~also~~ used in loops.

Break

→ Break the current iteration & also the remaining iterations

→ int i;

```
for(i=1; i<=20; i++)
{
    if(i==5)
    break;
    printf("%d\n", i);
}
```

⇒ O/P: ~~1 2 3~~
1
2
3
4

continue

→ Break the current iteration & ~~also~~ continue with the next iteration

→ Int i;

```
for (i=1; i<=20, i++)
{
    if (i==5)
        continue;
    printf("%d\n", i);
}
```

⇒ O/P:
1
2
3
4 → 5 is skipped
6
7
8
9
10
11
12

Q) write a C program to print all even numbers between 1 to 50?

Sol

```
#include <stdio.h>
int main()
{
 int i;
 for(i=1;i<=50;i++)
 {
     if(i%2!=0)
      continue;
     printf("%d\n",i);
 }
```

: Op:

2
4
6
8
:
:
50.

=> Arrays:

→ It is a collection (more number of elements in it)

→ It is type specific.

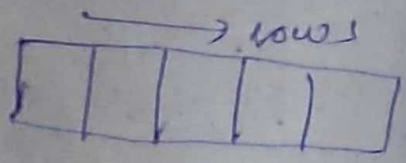
⇒ Types of arrays:

i) one dimensional arrays (1D-arrays)

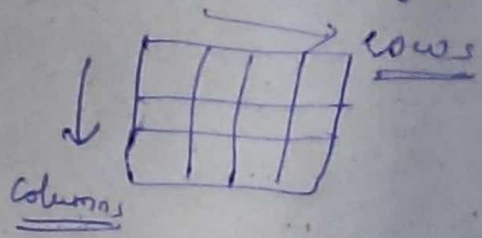
ii) Two dimensional arrays (2D-arrays)

iii) multi dimensional arrays

i) one dimensional arrays:



ii) Two-dimensional arrays.



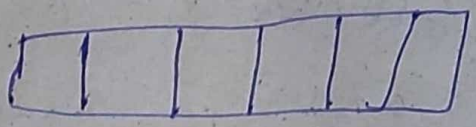
i) One Dimensional Arrays:

a) Declaration of 1D-array (Giving na

b) Definition (Specify type)

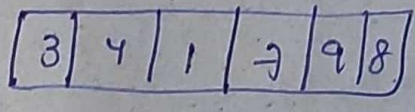
c) Initialization (Assigning value)

int arr[6];



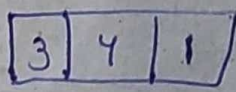
a) Basic initialization

int arr[6] = {3, 4, 1, 7, 9, 8};

arr → 

b) Initialization without size:

int arr[ ] = { 3, 4, 1};



13/1/22

⇒ Manipulation: (#include <string.h>)

i) String length

Engineering [10]

To calculate the length of string we use

⇒ Strlen( ) [function]

Syntax:

~~int Strlen(char);~~

int strlen (char[]);

↑ length

↑ string

Code:

```
#include<stdio.h>
#include<string.h>
int main(void)
{
   char str[] = "Engineering";
   int len;
   len = strlen(str);
printf("String length : %d", len);
}
```

2) String reverse:

Syntax: ~~Str rev~~ strrev( ); (function)

input

strrev(char[]);

string

Program:

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

3) String copy:

a) Basic String copy

```
str1[ ] = "hyderabad"
str2[ ];
```

Syntax: (str copy into str2)

```
strcpy(str2, str1);
```

b) String copy length controlled

```
str1[ ] = "hyderabad"
str2[ ];
```

Syntax:

~~strncpy(~~

```
strncpy(deststr, src str1, no. of char)
strncpy(char[], char[], int);
```

Example:

```
strncpy(str2, str1, 3);
puts(str2);
```

o/p: Hyd.

4) String Comparison: strcmp();

match

i) No. of character

"HELLO"] Equal ii) upper case

"HELLO"] iii) individual characters

---

5) ~~Basic String Comparison~~

→ Comparison of string has two possibilities equal and not equal.

→ If it is equal (zero) it gives +ve value

→ If it is not equal (non-zero) it gives +ve(or) -ve value

str1 ⟶ "hydᵉrabad";

str2 ⟶ "hydarabed";

Not equal (non-zero)

+ve, -ve.

→ According to alphabetical order

a b c d e ...

position of e is greater than a.

→ i) $str_1 > str_2$

e > a

we will get +ve

ii) $str_1 > str_2$

a < e

we will get -ve

→ If your 1st string is greater than 2nd string value we get +ve

→ otherwise -ve.

→ If they are equal we get '0'.

ASCII value.

toascii();

Syntax:

toascii(char);

→ It will give the ascii value of character.

Syntax:

strcmp

1) int strcmp(char[], char[]);

example str1 = "hyderabad";

str2 = "hydquabed";

int status;

status = strcmp(str1, str2);

↓ 0 → equal.

status = strcpy(str1, str2);

↓ +ve → e > a

status = strcpy(str2, str1);

↓ -ve → a < e

program:

i)
```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str1[] = "hyderabad";
    char str2[] = "hyderabad";
    int status;
    status = strcmp(str1, str2);
    printf("status = %d", status);
}
```

output: status = 0

ii) If
```
char str1[] = "hyderabad";
char str2[] = "hydarabad";
```
e > a

output: +1

iii)
```
char str1[] = "hyderabad";
char str2[] = "hyderabed";
```
a < e

output: -1

5) String concatenation (joining string together)

"hello" "world"

"helloworld"

a) Basic string concatenation.

syntax: strcat(char[], char[]);

It will join 2 string and store it in str1.

eg: str1 → "hydeabad";
str2 → "engineering";

strcat(str1, str2);

result: str1 → "hydeabad engineering"

b) string concatenation length controlled.

syntax: strncat(char[], char[], int);

No. of char from 2nd string to join.

Ex1: ~~strncat(char[], char~~

strncat(str1, str2, 3);

result → "hydeabad eng".

program:

```
#include <stdio.h>
#include <string.h>
int main(void)
{
  char str1[] = "hyderabad";
  char str2[] = "Engineering";
  strcat(str1, str2);
  puts(str1);
}
```

O/P: Hyderabad engineering.

If str_2 is put instead of str_1.

O/P: Engineering.

If if '3' is specified in strncat

O/P: "Hyderabad Eng".