



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

21CSS101J – Programming for Problem Solving

Unit II





SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

UNIT II

Conditional Control -Statements :Simple if, if...else - Conditional Statements : else if and nested if - Conditional Statements : Switch case - Un-conditional Control Statements : break, continue, goto - Looping Control Statements:for, while, do..while - Looping Control Statements: nested for, nested while - Introduction to Arrays -One Dimensional (1D) Array Declaration and initialization - Accessing, Indexing and operations with 1D Arrays



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

UNIT II

Array Programs – 1D - Initializing and Accessing 2D Array,
Array Programs – 2D - Pointer and address-of operators -
Pointer Declaration and dereferencing, Void Pointers, Null
pointers, Pointer based Array manipulation



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements

- ☐ Also called as Conditional Statement
- ☐ Decides order of execution based on conditions
- ☐ Helps repeat a group of statements
- ☐ Modifies control flow of program
- ☐ Decision Making
- ☐ Branching



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

❑ *Types of Branching Statements*

- a) if statement
 - i. Simple if
 - ii. if...else statement
 - iii. nested if...else statement
 - iv. else...if statement
- b) switch statement
- c) goto statement



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

a) if statement

- ☐ Condition "True" - Statement block will be executed
- ☐ Condition "False" - Statement block will not be executed.
- ☐ **Variations**
 - i. Simple if
 - ii. if...else statement
 - iii. nested if...else statement
 - iv. else...if statement



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

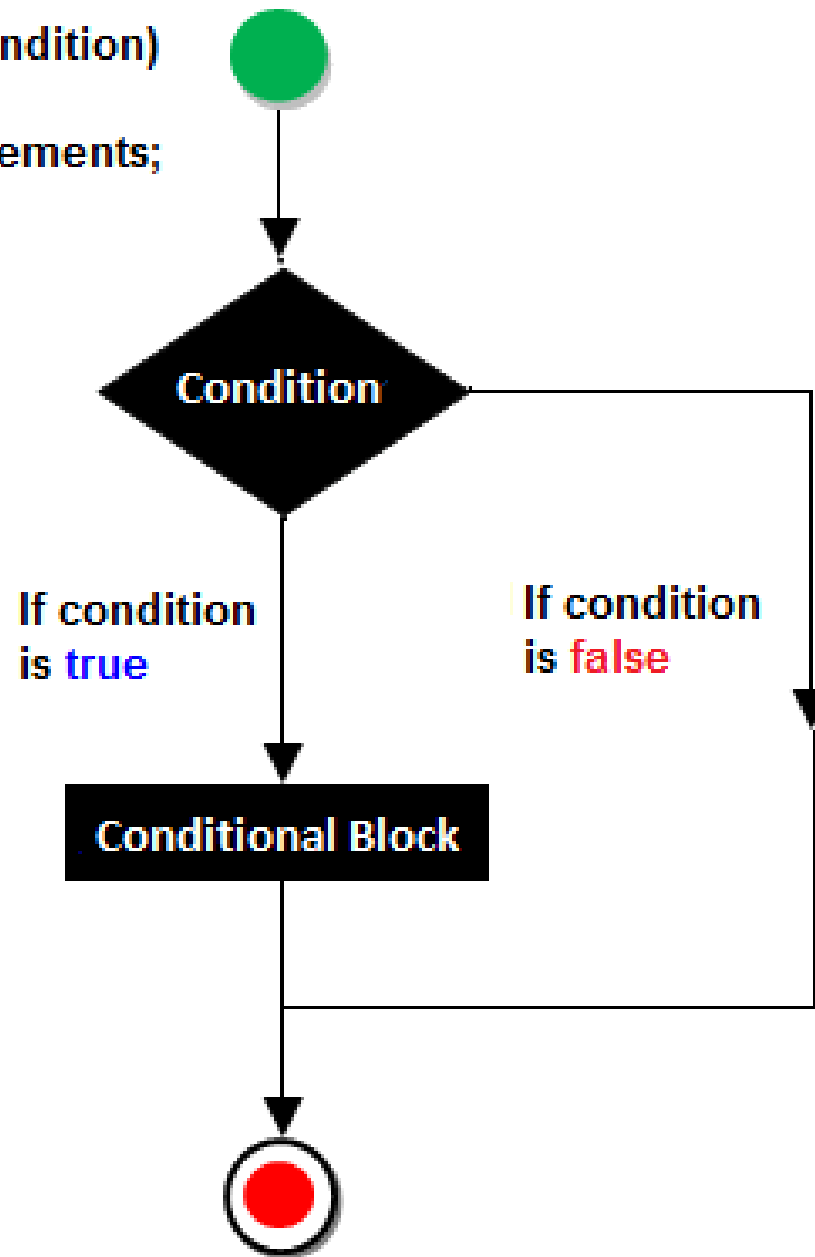
2. 1 Control Statements Contd...

i. Simple if statement

- ☐ Basic if statement
- ☐ What is a condition?
- ☐ Executes statement block only if condition is true
- ☐ **Syntax**

```
if (condition)  
{  
    Statements;  
}
```

```
if( condition)
{
  statements;
}
```



/ Simple if – Program to check whether a number is Odd*/*

```
#include<stdio.h>
int main( )
{
    int number;
    printf("Enter the Number:");
    scanf("%d", &number);
    if(number%2==0)
    {
        printf("The Number is Even");
    }
    return 0;
}
```

Output

Enter a value : 10342

The number is Even



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

❑ Try it Out Yourself! Write a C program to:

- 1) Check whether the given number is Odd
- 2) To check whether the given number is Greater
- 3) To check whether the given number is Smaller
- 4) To check whether the given number is positive
- 5) To check whether the given number is negative
- 6) To check whether the given number is zero
- 7) To check whether two numbers are equal



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

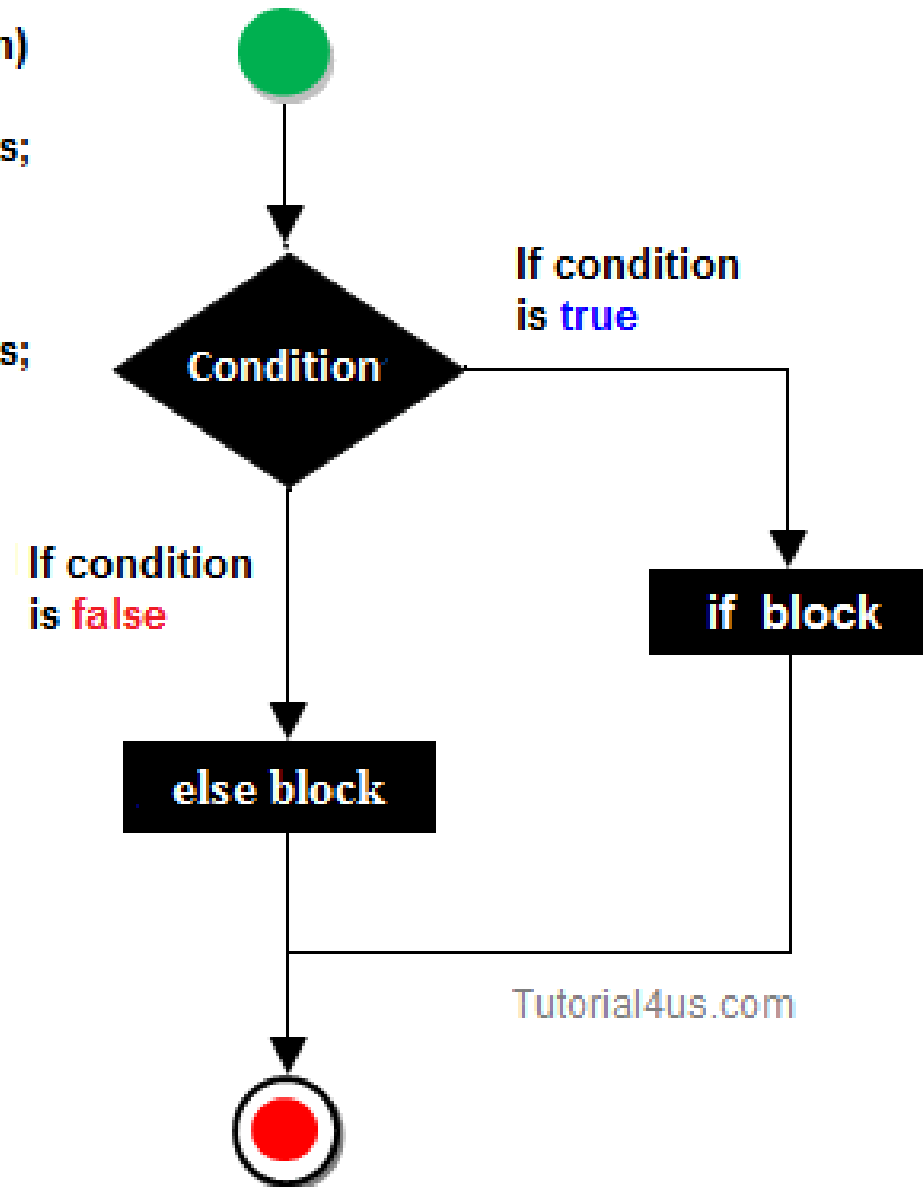
2. 1 Control Statements Contd...

ii. If else statement

- ☐ Extension of basic if statement
- ☐ Takes care of True and False condition
- ☐ Number of Statement Blocks - 2
 - ☐ Block 1 – True Condition
 - ☐ Block 2 – False Condition

```
if (condition)  
{  
    Statements;  
}  
Else  
{  
    Statements;  
}
```

```
if( condition)  
{  
    statements;  
}  
else  
{  
    statements;  
}
```



/ if else -Tocheck whether a number is Odd or Even*/*

```
#include<stdio.h>
int main( )
{
    int number;
    printf("Enter the Number: ");
    scanf("%d, &number);
    if(number%2==0)
    {
        printf("The Number is Even");
    }
    else
    {
        printf("The Number is Odd");
    }
    return 0;
}
```

Output 1

Enter the Number : 10341

The number is Odd

Output 2

Enter the Number : 10342

The number is Even



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

❑ Try it Out Yourself! Write a C program to:

- 1) To check whether the given number is Greater or Smaller
- 2) To check whether the given number is +ve or -ve
- 3) To check whether two numbers are equal or not



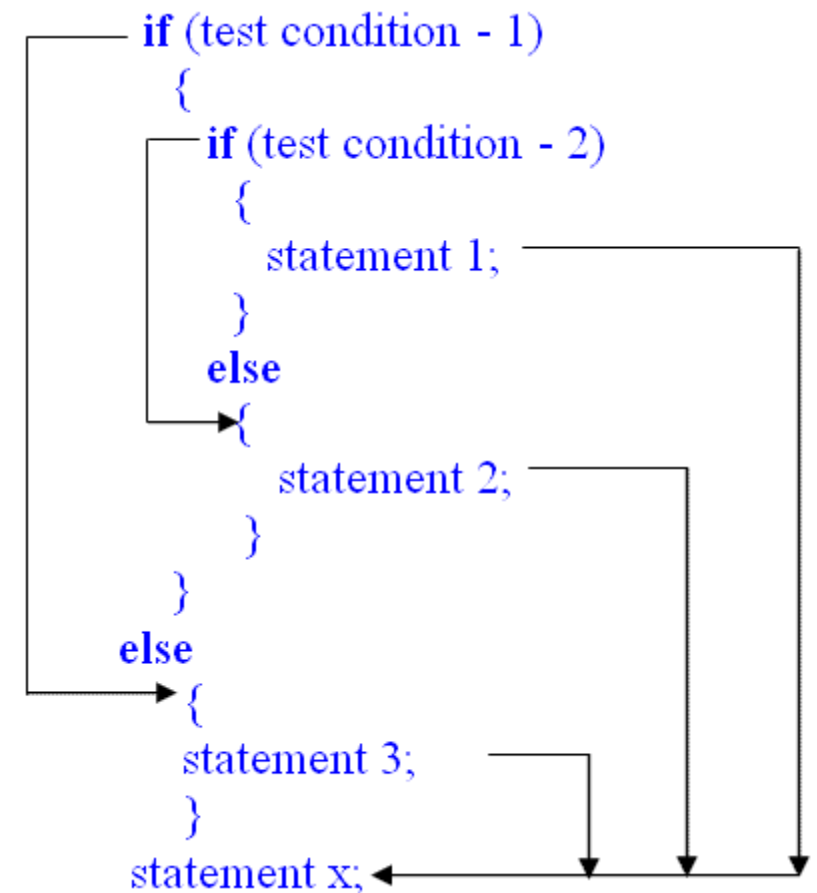
SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

iii. Nested if else statement

- ❑ Used when a series of decisions are involved
- ❑ Makes a choice between several alternatives
- ❑ New if else statement block is used within existing if else statement block



/*Program for Nested if else */

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

```
void main( )
```

```
{
```

```
    char username;
```

```
    int password;
```

```
    printf("Username:");
```

```
    scanf("%c",&username);
```

```
    printf("Password:");
```

```
    scanf("%d",&password);
```

```
if(username=='a')
{
    if(password==12345)
    {
        printf("Login successful");
    }
    else
    {
        printf("Password is incorrect, Try again.");
    }
}
else
{
    printf("Username is incorrect, Try again.");
}
return 0;
}
```

Output 1

Username: a

Password: 12345

Login Successful

Output 2

Username: a

Password: 54321

Password is incorrect, Try again.

Output 3

Username: b

Password: 54321

Username is incorrect, Try again.



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

- ❑ **Step 1:** First if condition will be true, if the user has typed 'a' as a username then the program control moves to second if condition and checks for the password
 - ❑ if it true it will print 'login successful'
 - ❑ else it will execute block statement 'Password is Incorrect, Try again.'
- ❑ **Step 2:** If the first if condition is false then it executes last else block thus printing 'Username is Incorrect, Try again.'

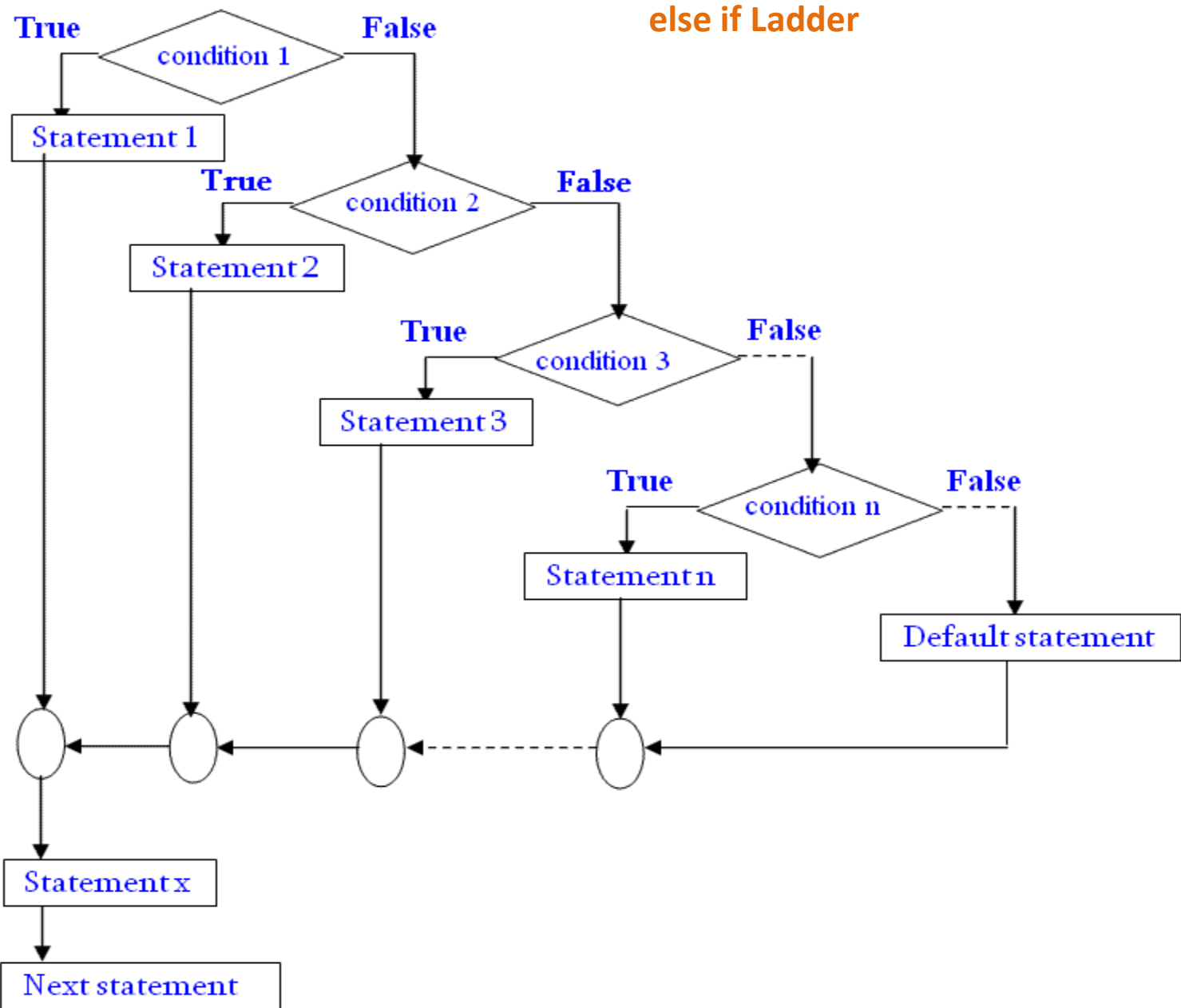


SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

- ❑ **Step 3:** In this above example we have use username as single character to use multiple character username we need to use string data type



*/*Program for if else ladder*/*

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a;
    printf("Enter a Number: ");
    scanf("%d",&a);
    if(a > 0)
    {
        printf("Given Number is Positive");
    }
    else if(a == 0)
    {
        printf("Given Number is Zero");
    }
    else if(a < 0)
    {
        printf("Given Number is Negative");
    }
    getch();
}
```



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

b) Switch statement

- ☐ Allows to make decisions from a number of choices
- ☐ Also called as Switch-Case-Default Statement
- ☐ Faster than nested if else statement
- ☐ Easier to understand
- ☐ **Rules for writing switch () statement**
 - ☐ Expression in switch must be an integer value or a character constant
 - ☐ No real numbers used in Expression



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

- ☐ Each case block and default block must end with break statements
- ☐ Default is optional
- ☐ Case keyword must end with colon (:)
- ☐ Default may be placed anywhere in the switch
- ☐ No two case constants are identical



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

```
switch(variable or expression)  
{  
    case constant 1:  
        statements;  
    break;  
    ....  
    case constant N;  
        statements;  
    break;  
    default:  
        statements;  
}
```

/ Program for Switch Case*/*

```
#include<stdio.h>
int main( )
{
    int a, b, choice;
    printf("\nEnter Two Numbers: ");
    scanf("%d%d", &a,&b);
    printf("\n Enter 1 for Addition");
    printf("\n Enter 2 for Subtraction");
    printf("\n Enter 3 for Multiplication");
    printf("\n Enter 4 for Division");
    printf(" Enter your Choice");
    scanf("%d",&choice);
```

```
switch (choice)
{
    case 1:
        printf("Sum is : %d", a+b);
        break;
    case 2:
        printf("Difference is : %d", a-b);
        break;
    case 3:
        printf("Multiplication is : %d", a*b);
        break;
    case 4:
        printf("Difference is : %d", a/b);
        break;
```

```
        default:
            printf("Invalid Choice:");

        }
    getch( );
}
```

Enter two numbers

20

10

Enter 1 for Addition

Enter 2 for Subtraction

Enter 3 for Multiplication

Enter 4 for Division

Enter your Choice: 3

Product is : 200



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

☐ **Nested Switch statement**

- ☐ Inner switch() can be a part of an outer switch()
- ☐ Inner switch() and outer switch() case constants may be the same

/ Program for Nested Switch Case*/*

```
#include<stdio.h>
int main( )
{
    int square, i, n, fact = 1,choice;
    printf("\n Enter Any Number: ");
    scanf("%d", &n);
    printf(" 1. Square \n");
    printf(" 2. Factorial \n");
    printf(" 3. Find Odd or Even \n");
    printf(" 4. Exit \n");
    printf(" Enter your Choice");
    scanf("%d", &choice);
```

```
switch (choice)
{
    case 1:
        square = n * n;
        printf("The Square of the Given number is %d\n",
square);
        break;
    case 2:
        for(i=1;i<=n;i++)
        {
            fact = fact * i;
        }
        printf("The Factorial of a given number is %d\n", fact);
        break;
```



```
case 3:
switch (n%2)
{
    case 0:
        printf("Given Number is Even\n");
    case 1:
        printf("Given Number is Odd\n");
}
case 4:
    exit(0);
default:
    printf("Invalid Choice. Please try again\n");
}
```

```
return 0;
```

```
}
```

Enter any number

5

1. Square
2. Factorial
3. Find Odd or Even
4. Exit

Enter your choice

2

The factorial of a given number is: 120



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 1 Control Statements Contd...

c) The goto statement

- ❑ Transfers control from one point to another

- ❑ **Syntax**

```
goto label;  
  
statements;  
  
.....  
  
label  
  
statements;
```



2.1 break

The break statement ends the loop immediately when it is encountered.

Its syntax is: break;

The break statement is almost always used with if...else statement inside the loop.

```
while (testExpression) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}  
↓  
do {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
} while (testExpression);  
↓  
for (init; testExpression; update) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}  
↓
```



2.1 Continue

The continue statement skips the current iteration of the loop and continues with the next iteration.

Its syntax is: continue;

The continue statement is almost always used with the if...else statement.

```
while (testExpression) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```

```
do {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
} while (testExpression);
```

```
for (init; testExpression; update) {  
    // codes  
    if (testExpression) {  
        continue;  
    }  
    // codes  
}
```



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements

- ❑ Loop – A segment of the program that is executed repeatedly until a condition is satisfied
- ❑ Classification – Entry Controlled & Exit Controlled
- ❑ **Types**
 - a) while do loop
 - b) do while loop
 - c) for loop
 - i. Nested for loop



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

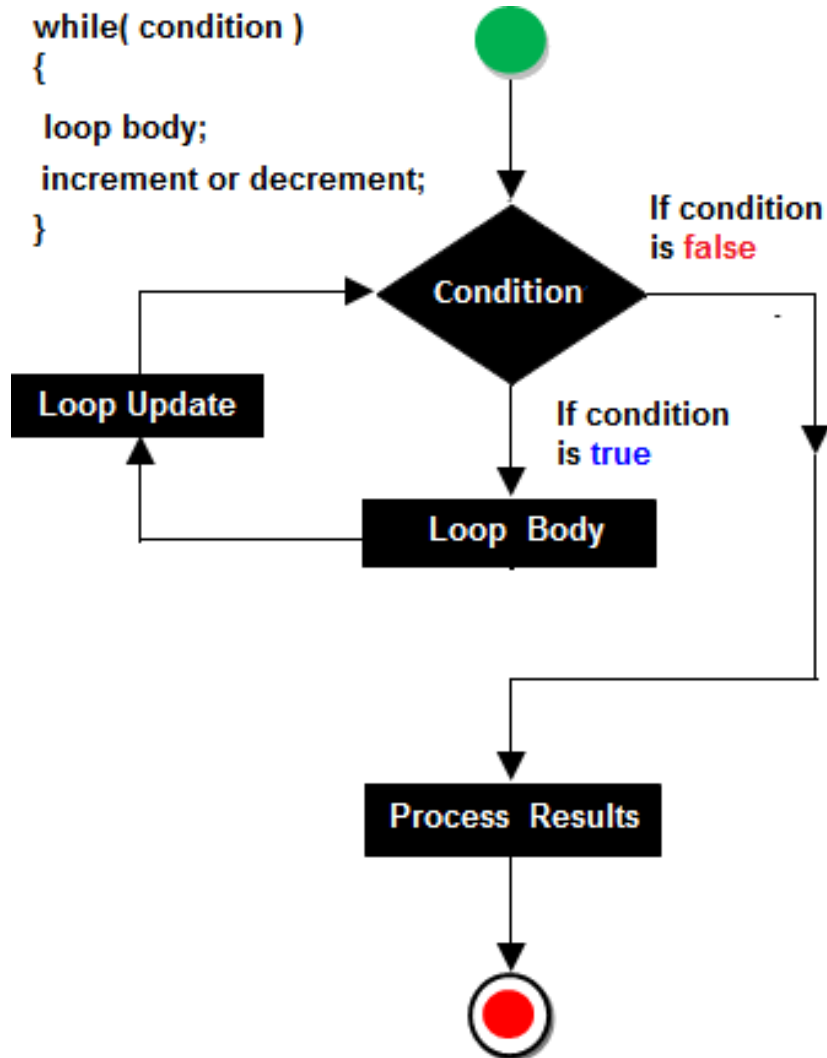
a) The While Loop

- ☐ Simplest looping structure in C
- ☐ Statements in the program may need to repeat for many times. e.g., calculate the value of $n!$
- ☐ Loop consists of two segments
 - ☐ Control Statement
 - ☐ Body of the Loop
- ☐ How while loop works?



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.



Initialize loop counter variable;

while (condition)

{

Statements;

increment / Decrement loop
counter variable;

}

/ Program to Add 3 Numbers*/*

```
#include<stdio.h>
int main( )
{
    int a, b, c, sum;
    printf("\n Enter the Three Numbers: ");
    scanf("%d%d%d", &a,&b,&c);
    sum = a+b+c;
    printf("The sum of 3 Numbers is %d", sum);
    return 0;
}
```

Output

Enter the Three Numbers: 10 20 30

The sum of 3 Numbers is: 60

/ Program to Add n Numbers*/*

```
#include<stdio.h>
int main( )
{
    int i=1,n, sum=0;
    printf("\n Enter the value for n: ");
    scanf("%d", &n);
    while (i<=n)
    {
        sum = sum + i;
        i++;
    }
    printf("The sum of n Numbers is: %d", sum);
    return 0;
}
```

Output

Enter the value for n: 5

The sum of n Numbers is: 15



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ **Try it Out Yourself ! Write a C program to:**

- 1) To print all even numbers from 1 to 100
- 2) To print all even numbers from 1 to n
- 3) To print table for any number
- 4) To calculate the sum of its digits
- 5) To check whether the entered number is Prime or not
- 6) To get a number as input and print it in reverse.
- 7) To check whether the number is Armstrong number



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

b) The Do While Loop

- ❑ The body of the loop is executed at least once

- ❑ **Syntax**

```
do
{
    statements;
}
while (condition);
```

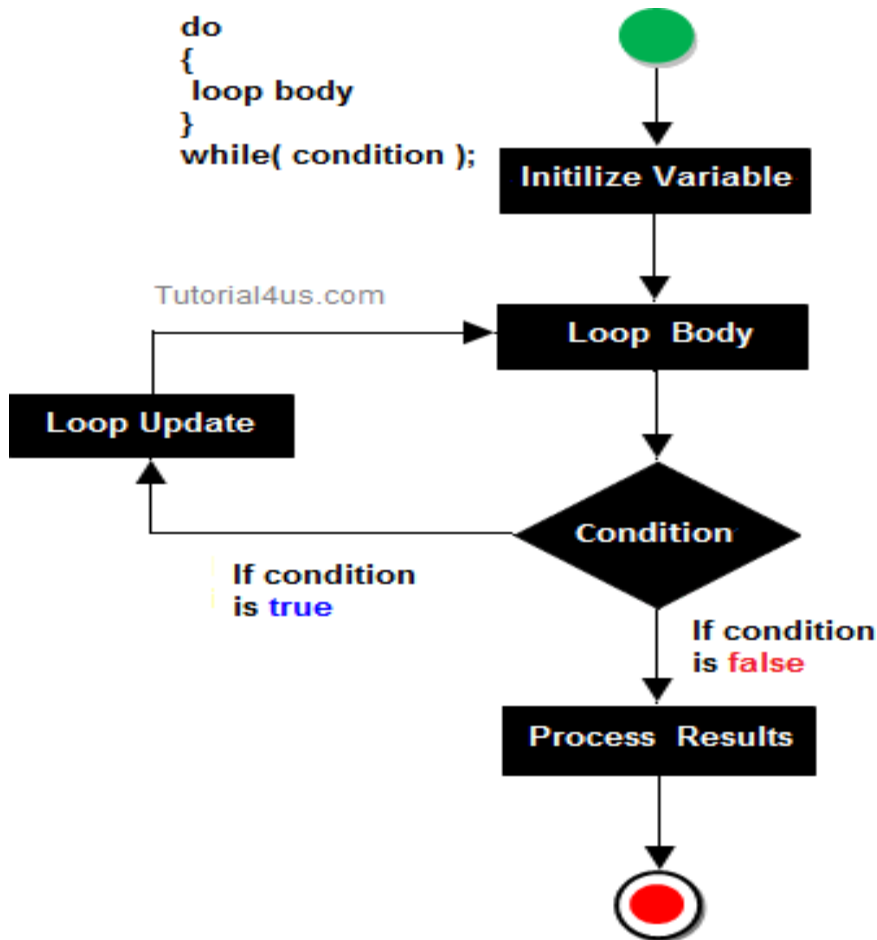


SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

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

Tutorial4us.com



Initialize loop counter variable;

do

{

Statements;

increment / Decrement loop
counter variable;

}

while (condition)



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

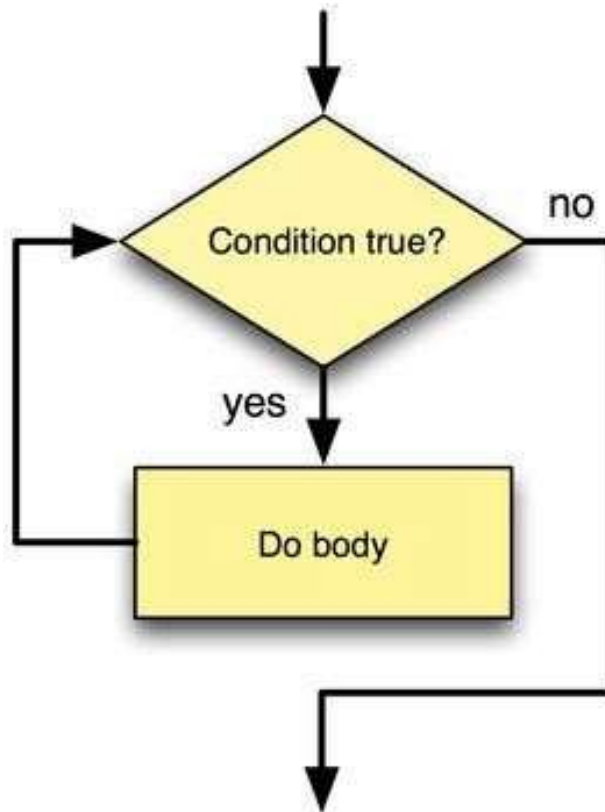
While Do Loop	Do While Loop
Entry Controlled Loop	Exit Controlled Loop
Test condition is checked before body of the loop is executed	Test condition is checked after the body of the loop is executed
Loop will not be executed if condition is false	Loop will be executed at least once even if condition is false
Top tested loop	Bottom tested loop



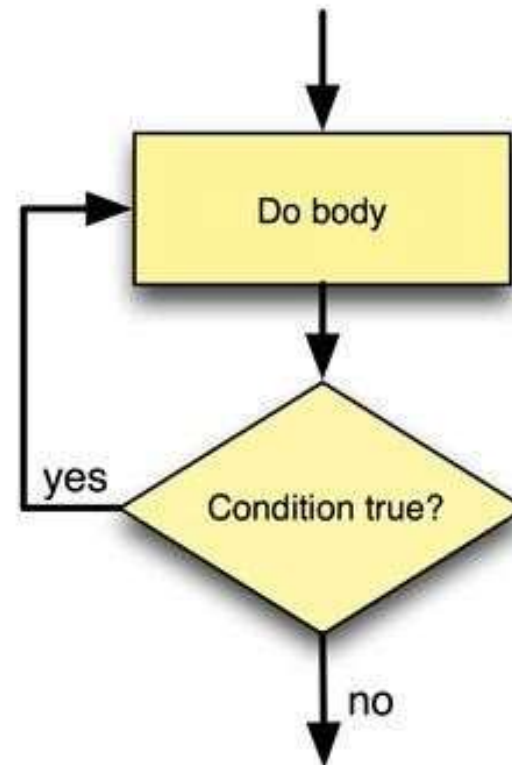
SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...



while flowchart



do/while flowchart



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

c) The for loop

- ☐ Most commonly and popularly used loop structure
- ☐ Structure of the for loop
 - ☐ Initialize loop counter variable
 - ☐ Check for condition
 - ☐ Increment / Decrement the loop counter variable
- ☐ **Syntax**

for(initialization; condition; increment / decrement)

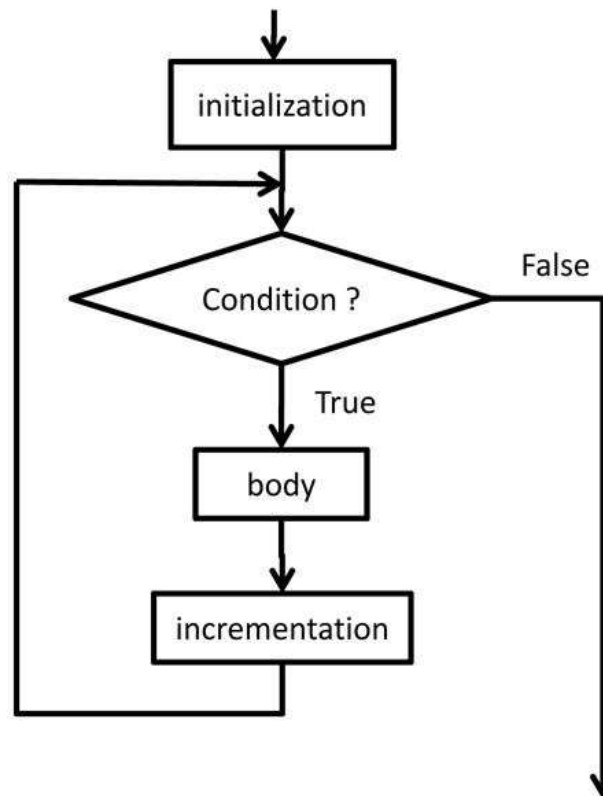


SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

for(initialization; condition; incrementation)
body;





SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ Examples

- i. for(i = 0; i < n; i++)
 {
 Statements;
 }
- ii. for(count = 0; count > n; count--)
 {
 Statements;
 }

/ Program to Add n Numbers using for loop */*

```
#include<stdio.h>
int main( )
{
    int i, n, sum=0;
    printf("\n Enter the value for n: ");
    scanf("%d", &n);
    for (i =1; i<=n; i++)
    {
        sum = sum + i;
    }
    printf("The sum of n Numbers is: %d", sum);
    return 0;
}
```

Output

Enter the value for n: 5

The sum of n Numbers is: 15



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

- ❑ nested for syntax

```
for (initialization; condition; increment/decrement)
{
    for (initialization; condition;
        increment/decrement)
    {
        body of the loop;
    }
}
```



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ nested for Example Program

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

```
#include<conio.h>
```

```
main()
```

```
{
```

```
Int row, column;
```

```
for(row=1;row<=5;row++)
```

```
{
```

```
for(column=1;column<=row;column++)
```

```
{
```



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ nested for Example Program

```
print("*");  
}  
print("\n");  
}  
getch(); }
```

Output:

```
*  
  
**  
  
***  
  
****  
  
*****
```

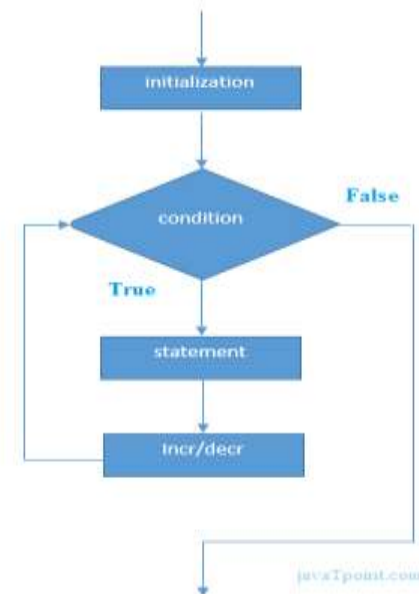

for loop in C

- The **for loop in C language** is used to iterate the statements or a part of the program several times.
- It is frequently used to traverse the data structures like the array and linked list.

Syntax of for loop in C

The syntax of for loop in c language is given below:

```
for(Expression 1; Expression 2; Expression 3){  
    //code to be executed  
}
```



Example for for loop:
/*simple program of for loop
that prints table of 1.*/

```
#include<stdio.h>
int main(){
int i=0;
for(i=1;i<=10;i++)
{
printf("%d \n",i);
}
return 0;
}
```

Output

```
1
2
3
4
5
6
7
8
9
10
```

/* Program to print
table of given
number*/

```
#include<stdio.h>
int main(){
int i=1,number=0;
```

```
printf("Enter a num  
ber: ");
```

```
scanf("%d",&numb  
er);
```

```
for(i=1;i<=10;i++){
```

```
printf("%d \n",(num  
ber*i));
```

```
}
```

```
return 0;
```

```
}
```

Output

```
Enter a number: 50
50
100
150
200
250
300
350
400
450
500
```

for loop in C

Properties of Expression 1

- The expression represents the initialization of the loop variable.
- We can initialize more than one variable in Expression 1.
- Expression 1 is optional.

Example 1

```
#include <stdio.h>

int main()
{
    int a,b,c;
    for(a=0,b=12,c=23;a<2;a++)
    {
        printf("%d ",a+b+c);
    }
}
```

Output

35 36

Example 2

```
#include <stdio.h>

int main()
{
    int i=1;
    for(;i<5;i++)
    {
        printf("%d ",i);
    }
}
```

Output

1 2 3 4

for loop in C

Properties of Expression 2

- Expression 2 is a conditional expression. It checks for a specific condition to be satisfied. If it is not, the loop is terminated.
- Expression 2 can have more than one condition. However, the loop will iterate until the last condition becomes false. Other conditions will be treated as statements.
- Expression 2 is optional.

```
#include <stdio.h>

int main()
{
    int i,j,k;
    for(i=0,j=0,k=0;i<4,k<8,j<10;i++)
    {
        printf("%d %d %d\n",i,j,k);
        j+=2;
        k+=3;
    }
}
```

Output

```
0 0 0
1 2 3
2 4 6
3 6 9
4 8 12
```

for loop in C

Properties of Expression 3

- Expression 3 is used to update the loop variable.
- We can update more than one variable at the same time.
- Expression 3 is optional.

```
#include <stdio.h>

void main ()
{
    int i=0,j=2;
    for(i = 0;i<5;i++,j=j+2)
    {
        printf("%d %d\n",i,j);
    }
}
```

```
0 2
1 4
2 6
3 8
4 10
```

Nested Loops in C

- C supports nesting of loops in C.
- **Nesting of loops** is the feature in C that allows the looping of statements inside another loop.
- Any number of loops can be defined inside another loop, i.e., there is no restriction for defining any number of loops.
- The nesting level can be defined at n times.
- You can define any type of loop inside another loop; for example, you can define '**while**' loop inside a '**for**' loop.

Nested for loop

The nested for loop means any type of loop which is defined inside the 'for' loop.

```
for (initialization; condition; update)
{
    for(initialization; condition; update)
    {
        // inner loop statements.
    }
    // outer loop statements.
}
```

Example of nested for loop

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

```
int main()
```

```
{
```

```
    int n;
```

```
    printf("Enter the value of n :");
```

```
    scanf("%d",&n);
```

```
    // Displaying the n tables.
```

```
    for(int i=1;i<=n;i++) // outer loop
```

```
    {
```

```
        for(int j=1;j<=10;j++) // inner loop
```

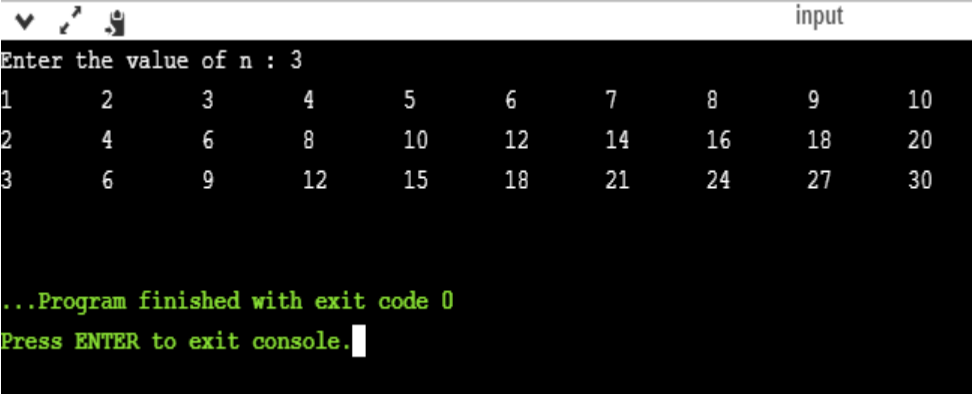
```
        {
```

```
            printf("%d\t",(i*j)); // printing the value.
```

```
        }
```

```
        printf("\n");
```

```
    }
```



```
input
Enter the value of n : 3
1      2      3      4      5      6      7      8      9      10
2      4      6      8      10     12     14     16     18     20
3      6      9      12     15     18     21     24     27     30

...Program finished with exit code 0
Press ENTER to exit console.
```


Explanation of the previous code

- First, the 'i' variable is initialized to 1 and then program control passes to the $i \leq n$.
- The program control checks whether the condition ' $i \leq n$ ' is true or not.
- If the condition is true, then the program control passes to the inner loop.
- The inner loop will get executed until the condition is true.
- After the execution of the inner loop, the control moves back to the update of the outer loop, i.e., $i++$.
- After incrementing the value of the loop counter, the condition is checked again, i.e., $i \leq n$.
- If the condition is true, then the inner loop will be executed again.
- This process will continue until the condition of the outer loop is true.



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ nested while loop

- Using while loop within while loop is said to be nested while loop.
- It is used to execute a set of code for multiple times as long as the condition is true.
- Once the condition becomes false the code block is no longer executed.



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ Syntax

```
while(test condition)
{
    while (test condition)
    {
        Body of the inner loop;
    }
    Body of the outer loop;
}
```

Nested while loop

```
#include <stdio.h>
int main()
{
    int i=1,j;
    while(i<=10){
        j=1;
        while(j<=10){
            printf("*");
            j++;
        }
        printf("\n");
        i++;
    }
    return 0;
}
```

* * * * *

* * * * *

* * * * *

* * * * *

* * * * *

* * * * *

* * * * *

* * * * *

* * * * *

* * * * *



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 2 Looping Statements Contd...

❑ **Try it Out Yourself ! Write a C program to:**

- 1) To print all even numbers from 1 to 100
- 2) To print all even numbers from 1 to n
- 3) To print table for any number
- 4) To calculate the sum of its digits
- 5) To check whether the entered number is Prime or not
- 6) To get a number as input and print it in reverse.
- 7) To check whether the number is Armstrong number



SRM

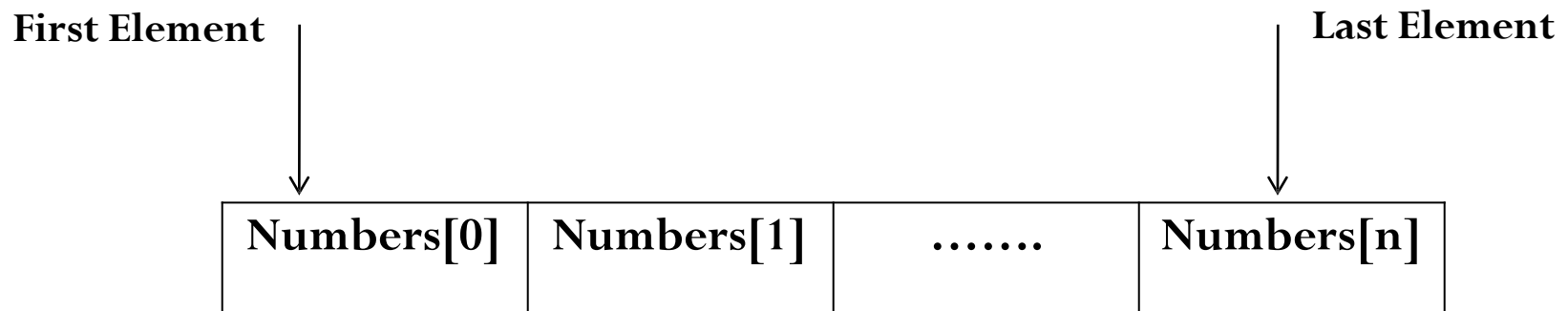
INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays

□ Definition

An array is defined as **finite ordered collection of homogenous** data, stored in contiguous memory locations.

- ✓ Array is used to store a collection of data
- ✓ Array is a collection of variables of the same type.





SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

☐ Need for Arrays

- ☐ Used to represent a list of numbers / names
- ☐ Used to represent tabular data in 2, 3 or more dimensions
- ☐ Important Data Structure in any programming language

☐ **Definition**

- ☐ Collection of elements of similar data types
- ☐ Each element is located in separate memory locations
- ☐ Each Array element share a common name



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

☐ Characteristics of Arrays

- ☐ All elements in the arrays share a common name
- ☐ Elements distinguished by index number
- ☐ Index (or) element number of an array plays vital role for calling each element
- ☐ Specific array elements can be modified
- ☐ Value of array element can be assigned to variables
- ☐ Array elements stored in continuous memory locations



SRM

**INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.**

2. 3 Arrays Contd...

- ❑ Storage space for array depends on its data type and size

Total bytes = sizeof (Data type) x Size of Array

- ❑ Example

`int a [5];`

Total bytes = sizeof (int) x 5 = 2 x 5 = 10 bytes



SRM

**INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.**

2. 3 Arrays Contd...

a) Array Declaration

☐ Syntax

Datatype arrayname [size/subscript];

☐ **Data Type:** int, float, double, char, structure, union

☐ **Array Name:** Name given to the Array variable

☐ **Size / Subscript:** Number of values an Array can hold

☐ **Examples**

int numbers[5];

float marks[50];

char name[20];

double a[i];



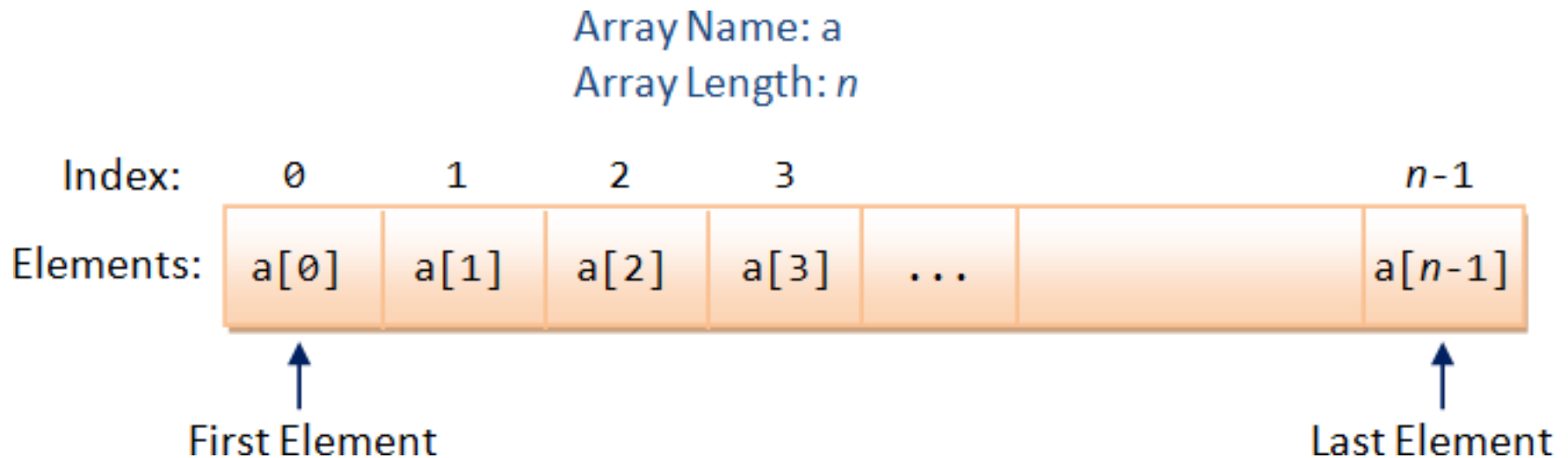
SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.

2. 3 Arrays Contd...

❑ Illustration

`int a[n];`





SRM

**INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.**

2. 3 Arrays Contd...

- ❑ **Static Array:** Array size (range) declared in the program
- ❑ **Dynamic Array:** Array size given during execution

STATIC ARRAYS	DYNAMIC ARRAYS
Range / Size of an array included in the Array definition	Range / Size of an array not included in the Array definition
Static Arrays cannot be changed	Dynamic Arrays can be changed



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

b) Array Initialization

- ❑ **Initialization:** Assigning values to array elements
 - ❑ Values specified in curly braces separated by commas
 - ❑ Examples

```
int a[ 5] = {1, 2, 3, 4, 5};
```

```
float b[3] = { 40.5, 59.0, 98.5};
```

```
char name[6] = " SRMIST";
```

- ❑ Array element index start from 0



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

- ❑ Array elements are called by array names followed by the element numbers
- ❑ `int a[5] = {1, 2, 3, 4, 5};`
 - `a[0]` refers to 1st element i.e., 1
 - `a[1]` refers to 2nd element i.e., 2
 - `a[2]` refers to 3rd element i.e., 3
 - `a[3]` refers to 4th element i.e., 4
 - `a[4]` refers to 5th element i.e., 5



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

c) Getting Input for Arrays

- ❑ Use for loops to get input in arrays
- ❑ Use for loops with regard to the Array's dimension
 - ❑ Input for One Dimensional Arrays – 1 for
loop for(i = 0; i < 5; i++)
{
 scanf("%d", &a[i]);
}



SRM

**INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.**

2. 3 Arrays Contd...

- ❑ Input for Two Dimensional Arrays – 2 for loops

```
for(i=0;i<5;i++)  
{  
    for(j=0;j<5;j++)  
    {  
        scanf("%d",&a[i][j]);  
    }  
}
```




SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

d) Printing Output in Arrays

- ☐ Use for loops to print array output
- ☐ Use for loops with regard to the Array's dimension
 - ☐ Printing One Dimensional Array Output – 1 for
loop for(i=0;i<5;i++)

 {

 printf(“%d”,a[i]);

 }



SRM

**INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.**

2. 3 Arrays Contd...

- ❑ Printing Two Dimensional Array Output – 2 for loops

```
for(i = 0; i < 5; i++)  
{  
    for(j=0; j < 5; j++)  
    {  
        printf("%d", a[i][j]);  
    }  
}
```

/* Program 1 : Array Declaration & Initialization*/

```
#include<stdio.h>
int main( )
{
    int i, arr[5];
    arr[0] = 10;
    arr[1] = 20;
    arr[2] = 30;
    arr[3] = 40;
    arr[4] = 50;
    for(i=0; i<=n; i++)
    {
        printf("%d"\n, a[i]);
    }
    return 0;
}
```

Output

10

20

30

40

50

/* Program 2 : Array Declaration & Initialization*/

```
#include<stdio.h>
int main( )
{
    int i, arr[5];
    arr[5] = {10, 20, 30, 40, 50};
    for(i=0; i<=n; i++)
    {
        printf("%d", a[i]);
    }
    return 0;
}
```

Output

10

20

30

40

50

/* Program 3 : Array Declaration & Initialization*/

```
#include<stdio.h>

int main( )
{
    int i, n, arr[5];
    scanf("%d", &n);
    printf("Enter the Elements of Array\n");
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("The Elements of the Array are\n");
```

```
    for(i=0; i<n; i++)  
    {  
        printf("%d", a[i]);  
    }  
    return 0;  
}
```

Output

Enter the Elements of the Array

10 20 30 40 50

The Elements of the Array are

10 20 30 40 50



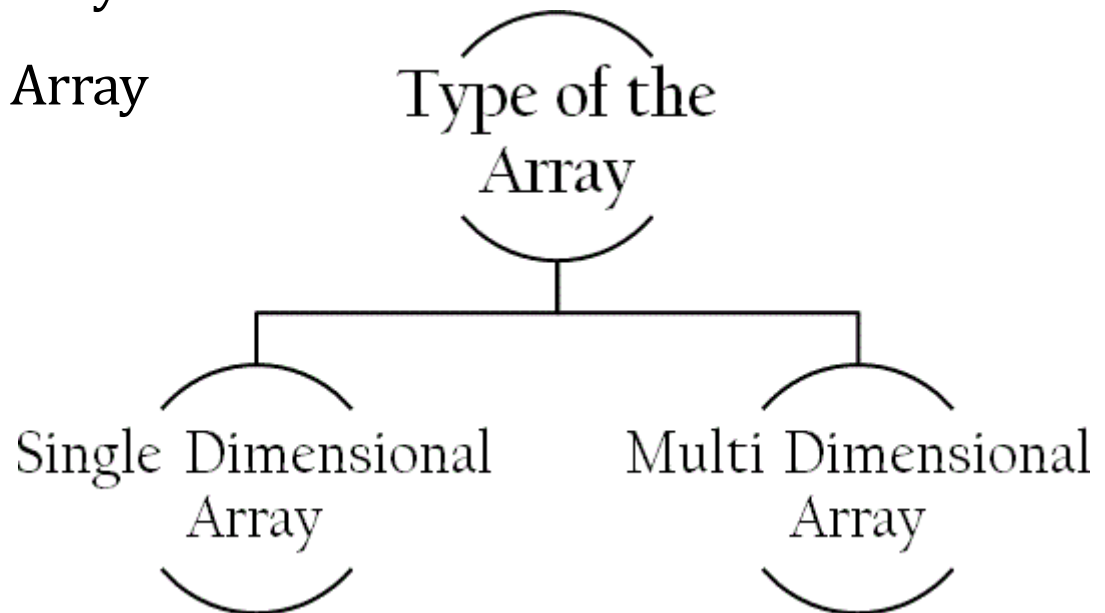
SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

e) Classification of Arrays

- i. One-Dimensional Array
- ii. Two-Dimensional Array
- iii. Multi-Dimensional Array





SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 3 Arrays Contd...

i. One Dimensional Array

- ❑ Data stored under a single variable using one subscript
- ❑ 1-D Array Declaration – Syntax

datatype arrayname [size/subscript];

❑ **Example:** int a [5];

- ❑ 1-D Array initialization – Syntax

datatype arrayname [size] = { list of values};

Example: int a [5] = { 10, 20, 30, 40, 50};

/* Program 1 : One Dimensional Array*/

```
#include<stdio.h>
int main ( )
{
    int a[10], n, i, sum;
    clrscr( );
    printf("Enter the Number of Elements\n");
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        scanf("%d", &a [i]);
    }
    sum = 0;
    for(i = 0; i < n; i++)
```

a [10]	
a [0]	40
a [1]	22
a [2]	34
a [3]	12
a [4]	64
a [5]	
a [6]	
a [7]	
a [8]	
a [9]	
n	i
5	0
sum	
0	

/ Program 1 : One Dimensional Array*/*

```
{  
    sum = sum + a[i];  
}  
printf("The Sum is: %d", sum);  
return 0;  
}
```

Output

Enter the Number of Elements

5

40 22 34 12 64

The Sum is 182

	a [10]
a [0]	40
a [1]	22
a [2]	34
a [3]	12
a [4]	64
a [5]	
a [6]	
a [7]	
a [8]	
a [9]	

n

5

i

4

sum

182

/ Program 2 : 1-D Array for Sorting*/*

```
#include<stdio.h>
int main( )
{
    int i, j, temp, n, a[10];
    printf("Enter the Number of Elements:");
    scanf("%d", &n);
    printf("Enter the Elements to be Sorted\n");
    for(i=0; i<n; i++)
    {
        scanf("%d\n", &a[i]);
    }
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
```

```
        if(a[i] > a[j])
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
}
print("The Sorted Elements are: \n");
for(i=0; i<n; i++)
{
    printf("%d\n", a[i]);
}
return 0;
}
```

Output

Enter the Number of Elements:5

Enter the Elements to be Sorted

25

12

45

68

7

The Sorted Elements are:

7

12

25

45

68



SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 4 - Initializing and Accessing 2D Array

An **array of arrays** is known as **2D array**. The two dimensional (2D) array in C programming is also known as matrix. A matrix can be represented as a table of rows and columns.

The syntax to declare the 2D array is

```
data_type array_name[rows][columns];
```




SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI.

2. 4 - Initializing and Accessing 2D Array

Initialization of 2D Array in C

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously.

However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.

```
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

/ Program 1 : 2-D Array */*

```
#include<stdio.h>

int main(){
int i=0,j=0;
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
//traversing 2D array
for(i=0;i<4;i++){
    for(j=0;j<3;j++){
        printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]);
    }//end of j
} //end of i
return 0;
}
```

Output

arr[0][0] = 1

arr[0][1] = 2

arr[0][2] = 3

arr[1][0] = 2

arr[1][1] = 3

arr[1][2] = 4

arr[2][0] = 3

arr[2][1] = 4

arr[2][2] = 5

arr[3][0] = 4

arr[3][1] = 5

arr[3][2] = 6

```

/* Program 2 : 2-D Array Storing elements in a matrix and printing it*/
#include <stdio.h>
void main ()
{
    int arr[3][3],i,j;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            printf("Enter a[%d][%d]: ",i,j);
            scanf("%d",&arr[i][j]);
        }
    }
    printf("\n printing the elements ....\n");
    for(i=0;i<3;i++)
    {
        printf("\n");
        for (j=0;j<3;j++)
        {
            printf("%d\t",arr[i][j]);
        }
    }
}

```

Output

Enter a[0][0]: 56

Enter a[0][1]: 10

Enter a[0][2]: 30

Enter a[1][0]: 34

Enter a[1][1]: 21

Enter a[1][2]: 34

Enter a[2][0]: 45

Enter a[2][1]: 56

Enter a[2][2]: 78

printing the elements

56 10 30

34 21 34

45 56 78

Pointers in C:

- Pointers in C are used to store the address of variables or a memory location.
- This variable can be of any data type i.e, int, char, function, array, or any other pointer.
- The size of the pointer depends on the architecture.
- **Syntax:**
- `datatype *var_name`

Uses of pointer

- To [pass arguments by reference](#)
- For [accessing array elements](#)
- To [return multiple values](#)
- [Dynamic memory allocation](#)
- To [Implement data structures](#)
- To do [System-Level Programming](#) where memory addresses are useful

Pointers in C:

How to Use Pointers?

To use pointers in C, we must understand below two operators.

1. To access the address of a variable to a pointer, we use the unary operator **&** (ampersand) that returns the address of that variable.

- For example **&x** gives us the address of variable **x**.

```
// The output of this program can be different
// in different runs. Note that the program
// prints address of a variable and a variable
// can be assigned different address in different
// runs.
```

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

```
int main()
```

```
{
```

```
    int x;
```

```
    // Prints address of x
```

```
    printf("%d", &x);
```

```
    return 0;
```

```
}
```

Output

0x7ffffd60dddfc

Pointers in C:

2. One more operator is **unary *** (Asterisk) which is used for two things:

2.A. To declare a pointer variable: When a pointer variable is declared in C/C++, there must be a * before its name.

// C program to demonstrate declaration of pointer variables.

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

```
int main()
```

```
{
```

```
    int x = 10;
```

```
    // 1) Since there is * in declaration, ptr becomes a  
    // pointer variable (a variable that stores address of  
    // another variable)
```

```
    // 2) Since there is int before *, ptr is pointer to  
    // an integer type variable
```

```
    int* ptr;
```

```
    // & operator before x is used to get address of x.
```

```
    // The address of x is assigned to ptr.
```

```
    ptr = &x;
```

```
    printf(“%d\n%d”,x,ptr);
```

```
    return 0;
```

```
}
```


Pointers in C:

2.B. To access the value stored in the address we use the unary operator (*) that returns the value of the variable located at the address specified by its operand. This is also called **Dereferencing**.

// C program to demonstrate use of * for pointers

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

```
int main()
```

```
{
```

```
    int Var = 10; // A normal integer variable
```

```
    int* ptr = &Var; // A pointer variable that holds address of var.
```

```
    // This line prints value at address stored in ptr.
```

```
    // Value stored is value of variable "var"
```

```
    printf("Value of Var = %d\n", *ptr);
```

```
    // The output of this line may be differen
```

```
    // runs even on same machine.
```

```
    printf("Address of Var = %p\n", ptr);
```

```
    // We can also use ptr as lvalue (Left han  
    // side of assignment)
```

```
    *ptr = 20; // Value at address is now 20
```

```
    // This prints 20
```

```
    printf("After doing *ptr = 20, *ptr is %d\n", *ptr);
```

```
    return 0;
```

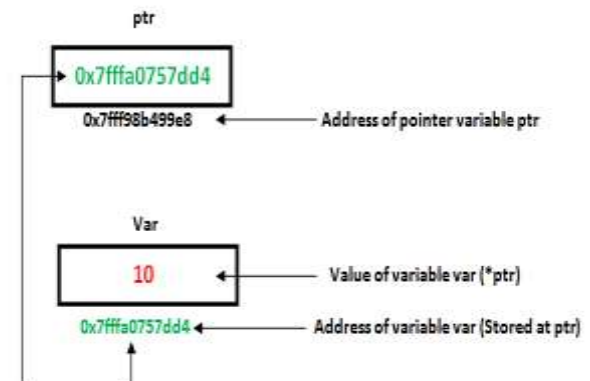
```
}
```

Output

Value of Var = 10

Address of Var = 0x7ffd11cd52ac

After doing *ptr = 20, *ptr is 20



Pointers in C:

Pointer Expressions and Pointer Arithmetic

A limited set of arithmetic operations can be performed on pointers.

The [Pointer Arithmetic](#) is slightly different from the ones that we generally use for mathematical calculations. The operations are:

- Increment/Decrement of a Pointer
- Addition of integer to a pointer
- Subtraction of integer to a pointer
- Subtracting two pointers of the same type
- Comparison of pointers of the same type.
- Pointer arithmetic is meaningless unless performed on an array.

Pointers in C:

Pointer Expressions and Pointer Arithmetic

// C program to illustrate Pointer Arithmetic

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

```
int main()  
{
```

```
    // Declare an array
```

```
    int v[3] = { 10, 100, 200 };
```

```
    // Declare pointer variable
```

```
    int* ptr;
```

```
    // Assign the address of v[0] to ptr
```

```
    ptr = v;
```

```
    for (int i = 0; i < 3; i++) {  
        printf("Value of *ptr = %d\n", *ptr);  
        printf("Value of ptr = %p\n\n", ptr);
```

```
        // Increment pointer ptr by 1
```

```
        ptr++;
```

```
    }
```

```
    return 0;
```

```
}
```

Output

Value of *ptr = 10

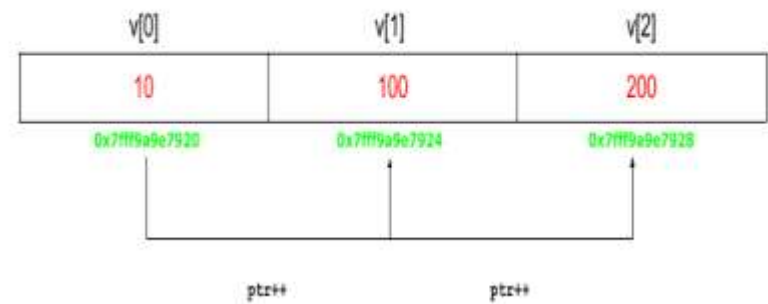
Value of ptr = 0x7ffe8ba7ec50

Value of *ptr = 100

Value of ptr = 0x7ffe8ba7ec54

Value of *ptr = 200

Value of ptr = 0x7ffe8ba7ec58



Pointers in C:

Pointers with one dimensional Arrays

- An array name acts like a pointer constant. The value of this pointer constant is the address of the first element.
- In the statement `int num[10];` the address of the first array element can be expressed as either `&num[0]` or `num`. Similarly, the address of the second array element can be written as `&num[1]` or as `num + 1`, and so on.
- In general, the address of the i^{th} array element can be expressed as either `&num[i]` or `(num + i)`.
- The expressions `&num[i]` and `(num + i)` both represent the address of the i^{th} element of `num`, and `num[i]` and `*(num + i)` both represent the value at that address

Pointers in C:

Pointers with one dimensional Arrays

```
#include <stdio.h>

void geeks()
{
    // Declare an array
    int val[3] = { 5, 10, 15 };

    // Declare pointer variable
    int* ptr;

    // Assign address of val[0] to ptr.
    // We can use ptr=&val[0];(both are same)
    ptr = val;

    printf("Elements of the array are: ");

    printf("%d, %d, %d",ptr[0],ptr[1],ptr[2]);

    return;
}

// Driver program
int main()
{
    geeks();
    return 0;
}
```

Output

Elements of the array are: 5 10 15

Val[0]	Val[1]	Val[2]
5	10	15
ptr[0]	ptr[1]	ptr[2]

Pointers in C:

Pointers and Multidimensional Arrays

Consider pointer notation for the two-dimensional numeric arrays.
consider the following declaration

```
int nums[2][3] = { {16, 18, 20}, {25, 26, 27} };
```

In general, `nums[i][j]` is equivalent to

Pointer Notation	Array Notation	Value
<code>*(*nums)</code>	<code>nums[0][0]</code>	16
<code>*(*nums + 1)</code>	<code>nums[0][1]</code>	18
<code>*(*nums + 2)</code>	<code>nums[0][2]</code>	20
<code>*(*(nums + 1))</code>	<code>nums[1][0]</code>	25
<code>*(*(nums + 1) + 1)</code>	<code>nums[1][1]</code>	26
<code>*(*(nums + 1) + 2)</code>	<code>nums[1][2]</code>	27

Pointers in C:

Dangling pointer

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer. There are **several** different ways where Pointer acts as dangling pointer

Example:

```
// Deallocating a memory pointed by ptr causes
```

```
// dangling pointer
```

```
#include <stdlib.h>
```

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

```
int main()
```

```
{
```

```
    int *ptr = (int *)malloc(sizeof(int));
```

```
    // After below free call, ptr becomes a
```

```
    // dangling pointer
```

```
    free(ptr);
```

```
    // No more a dangling pointer
```

```
    ptr = NULL;
```

```
}
```

Pointers in C:

NULL Pointers

Null pointers can be created by assigning a zero value during pointer declaration. This method is useful when no address is assigned to the pointer.

Syntax:

```
int * ptr = NULL;
```

```
// C program to show use of Null Pointer  
#include <stdio.h>
```

```
int main()  
{  
    // null pointer  
    int* ptr = NULL;  
  
    printf("The value inside variable ptr is:\n%x", ptr);  
  
    return 0;  
}
```


Pointers in C:

Wild Pointers

Wild Pointers are pointers that have not been initialized with something yet. These types of C-pointers can cause problems in our programs and can eventually cause them to crash. While working with Wild Pointers Programmer must be very careful.

// C Program to show use of wild pointers

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

```
int main()
```

```
{
```

```
    // wild pointer
```

```
    int* ptr;
```

```
    printf("\n%d", *ptr);
```

```
    return 0;
```

```
}
```

Pointers in C: void pointer

When a pointer is declared with a void keyword, then it is called a void pointer. To print the value of this pointer, you need to typecast it.

Syntax:

```
void *var;
```

Example:

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

```
int main()
```

```
{
```

```
    int a=2;
```

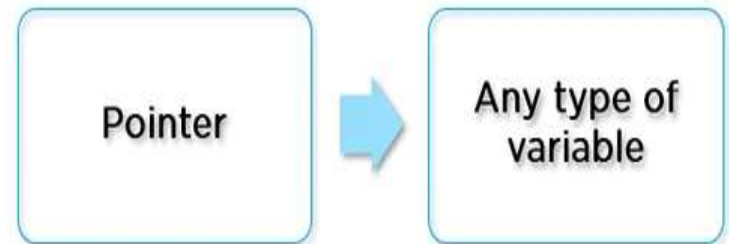
```
    void *ptr;
```

```
    ptr= &a;
```

```
    printf("After Typecasting, a = %d", *(int *)ptr);
```

```
    return 0;
```

```
}
```



```
After Typecasting, a = 2
```

```
...Program finished with exit code 0
```

```
Press ENTER to exit console.
```

Scenario based Question:

Chandran has to travel to another place. For this, he can avail any one of two cab services.

- The first cab service charges X rupees.
- The second cab service charges Y rupees.

He wants to spend the **minimum** amount of money. Which cab service should Chef take? Implement the logic using pointers.

Input Format

- The first line will contain T - the number of test cases. Then the test cases follow.
- The first and only line of each test case contains two integers X and Y - the prices of first and second cab services respectively.

Output Format

For each test case, output **FIRST** if the first cab service is cheaper, output **SECOND** if the second cab service is cheaper, output **ANY** if both cab services have the same price.

```
#include <stdio.h>
void main()
{
    int n, fcab, scab, *ptr1=&fcab, *ptr2=&scab;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", ptr1);
        scanf("%d", ptr2);
        if(*ptr1<*ptr2)
        {
            printf("\nFirst");
        }
        else if(*ptr2<*ptr1)
        {
            printf("\nSecond");
        }
        else
            printf("\nAny");
    }
}
```

THANK YOU