



AMBITION GURU

Compiled by ab

Computer Concept and Programming

I semester, BSc. CSIT



Syllabus

Unit	Contents	Hours
1.	Computer Fundamental and Programming Methodology	3
2.	Overview of C Language	4
3.	Control Structures	6
4.	Arrays and Strings	6
5.	Functions	8
6.	Pointers	5
7.	Structures and Unions	5
8.	File Handling in C	5
9.	Introduction to Graphics	3

Practical
Works

Credit hours : 3

Unit 3

(6 hrs.)

Compiled by ab

Control Structures

- Introduction,
- Decision Making Statements,
- Looping Statements,
- Branching Statements,
- Common Pitfalls: Infinite loops, Misplacement of Conditionals.

Compiled by ab

Unit 3: Part 1

Introduction, Decision Making Statements

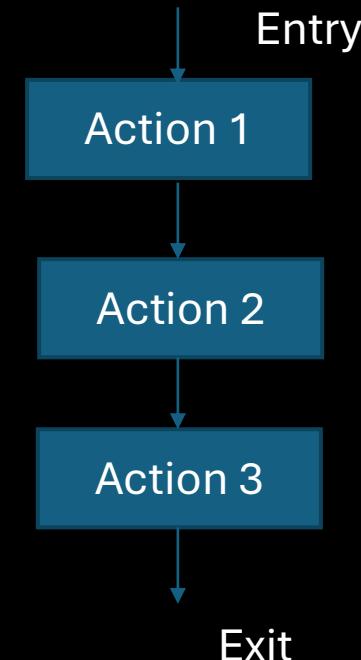
Control Structure

- Control structure enables us to specify the order in which the various instructions in a program are to be executed by the computer.
- In other words control instructions determine the “flow of control” in a program.
- There are 3 types of control instructions in C. They are:
 1. Sequential Control instruction
 2. Selection or decision control instruction
 3. Repetition or looping control instruction

Control Structure : Sequential Control Structure

Compiled by ab

- Instructions are executed in the same order in which they appear
- Each instruction is executed exactly once.
- No any condition is evaluated.



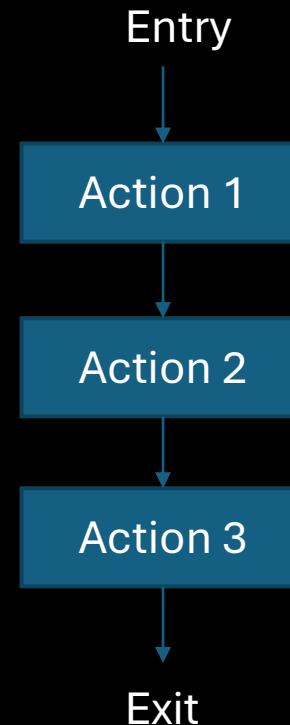
Control Structure : Sequential Control Structure

Example:

Compiled by ab

Program to find the sum of two numbers.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num1, num2, sum;
    printf("Enter the two numbers");
    scanf("%d%d", &num1, &num2);
    sum=num1+num2;
    printf("Sum of two numbers is %d", sum);
    getch();
    return 0
}
```





AMBITION GURU

Compiled by ab

Selection/Branching/Decision Control instructions



Selection/Branching/Decision Control instructions

Compiled by ab

- They are used when we have a number of situations where we may need to change the order of execution of statements based on certain conditions.
 - I. simple if statement
 - II. if else statement
 - III. nested if else statement
 - IV. else if ladder

Compiled by ab

Selection/Branching/Decision Control instructions : **if statement**

Selection/Branching/Decision Control instructions : if statement

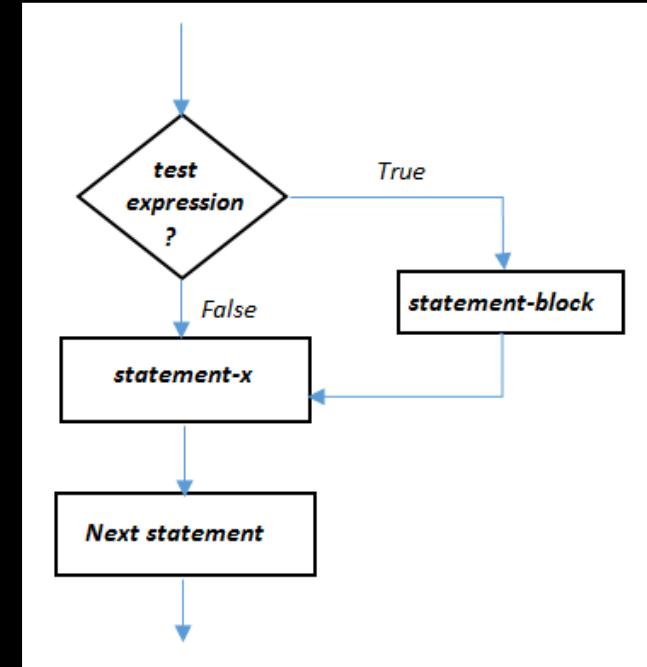
Compiled by ab

Evaluates the expression first then do the following:

- If the value of expression is true , it executes the statement within the block
- Otherwise it skips the statements within its block and continues from the first statement outside the block

Syntax:

```
if(test expression)
{
    statement-block;
}
statement-x;
```



if statement example

WAP that reads an integer from
the user and checks if the
number is positive.

```
#include <stdio.h>
Compiled

int main() {
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number);
    if (number > 0) {
        printf("The number is positive.\n");
    }
    return 0;
}
```



if statement example

WAP to input the average marks of a student and add 10% bonus marks if his/her average marks is greater than or equal to 65.

Compiled by ab

```
#include<stdio.h>
#include<conio.h>
int main()
{
    float marks;
    printf("Enter the marks\n");
    scanf("%f", &marks);
    if(marks>=65)
    {
        marks=marks+marks*0.1;
    }
    printf("Final marks=%f",marks);
    getch();
    return 0;
}
```

Compiled by ab

Selection/Branching/Decision Control
instructions : **if-else statement**



Selection/Branching/Decision Control instructions : if-else statement

Compiled by ab

Extension of simple if statement

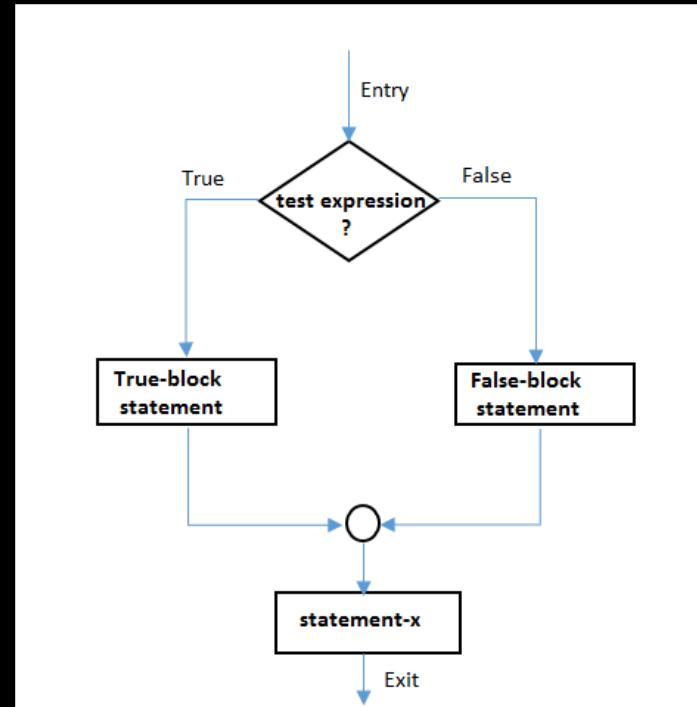
- If test expression is true,
 - then true block statement(s) (*immediately following the if statements are executed*)
- otherwise false block statements are executed

Selection/Branching/Decision Control instructions : if-else statement

Compiled by ab

Syntax:

```
if(test_expression)
{
    true-block statements
}
else
{
    false-block statements
}
statement-x;
```





Question

ed by ab

Write a C program that reads the age of a person and checks if they are eligible to vote. A person is eligible to vote if they are 18 years or older. If they are eligible, display a message indicating that they can vote; otherwise, display a message indicating that they are not eligible.



Question

```
#include <stdio.h>
ed by aman

int main() {
    int age;
    printf("Enter your age: ");
    scanf("%d", &age);

    if (age >= 18) {
        printf("You are eligible to vote.\n");
    } else {
        printf("You are not eligible to vote.\n");
    }

    return 0;
}
```

Compiled by ab

Selection/Branching/Decision Control instructions : **Nested if-else statement**



Selection/Branching/Decision Control instructions : Nested if-else statement

- When a series of decisions are involved, we may have to use more than one if. . . . else statement in nested form as shown below:

Syntax:

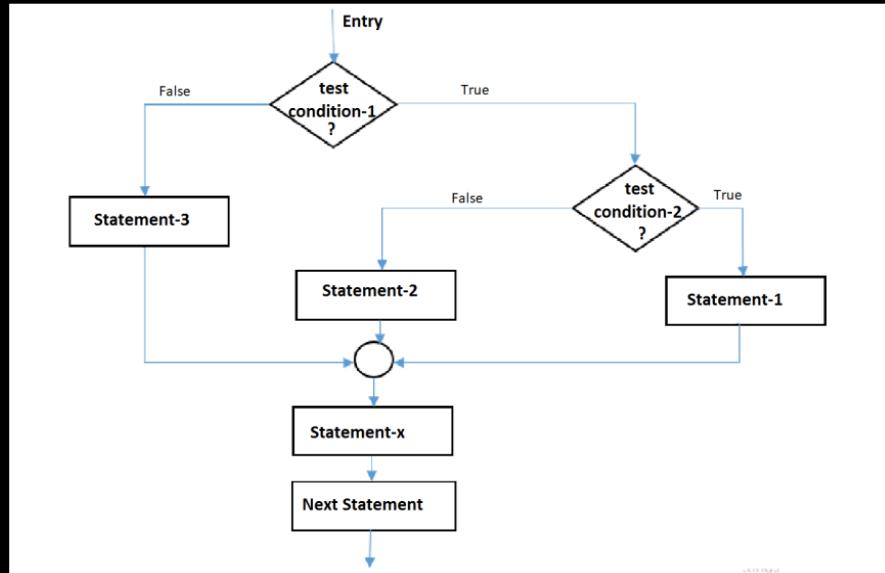
```
if(test condition-1)
{
    if(test condition-2)
    {
        Statement -1;
    }
    else
    {
        Statement-2;
    }
}
else
{
    Statement-3;
}
Statement x;
```

Selection/Branching/Decision Control instructions : Nested if-else statement

Syntax:

```
if(test condition-1)
{
    if(test condition-2)
    {
        Statement -1;
    }
    else
    {
        Statement-2;
    }
}
else
{
    Statement-3;
}
Statement x;
```

Compiled by ab





Nested if-else example

- WAP to find the largest number among three numbers

```
#include<stdio.h>
#include<conio.h>
int main() ab
Compiled ab
{
    int a,b,c;
    printf("Enter three number\n");
    scanf("%d%d%d", &a, &b, &c);
    if(a>b)
    {
        if(a>c)
        {
            printf("The largest number is %d",a);
        }
        else
        {
            printf("The largest number is %d",c);
        }
    }
}
```

Nested if-else example

- WAP to find the largest number among three numbers

```
Compiled by ab
else
{
    if(b>c)
    {
        printf("Largest number is %d",b);
    }
    else
    {
        printf("largest number is %d",c);
    }
}
```



Nested if-else example

- WAP to find the largest number among three numbers

```
else
{
    compiled by ab
    If(b>c)
    {
        printf("Largest number is %d",b);
    }
    else
    {
        printf("largest number is %d",c);
    }
}
```

Compiled by ab

Selection/Branching/Decision Control
instructions : **else-if ladder statement**

else-if ladder statement

- It is used when multiple decisions are involved. Here, the condition expression is evaluated in order.
 - ✓ If any of these expression is true, the statement associated with it is executed and this terminates the whole chain.
 - ✓ If none of the expression is true then statement associated with final else is executed.

Syntax:

```
if(condition-1)
{
    statement-1; _____
}
else if(condition-2)
{
    statement-2; _____
}
else if(condition-3)
{
    statement-3; _____
}
.....
.....
else if (condition-n)
{
    statement-n; _____
}
else
{
    default-statement; _____
}
statement-x: _____
```



else-if ladder statement

- Write a C program to read a person's marks and determine their grade using an **else-if ladder**. The grading criteria are as follows:

Marks \geq 90: Grade A

Marks \geq 80: Grade B

Marks \geq 70: Grade C

Marks \geq 60: Grade D

Marks $<$ 60: Fail

Compiled by AMBITION GURU

```
#include <stdio.h>
int main() {
    int marks;
    printf("Enter your marks: ");
    scanf("%d", &marks);
    if (marks >= 90) {
        printf("Grade: A\n");
    } else if (marks >= 80) {
        printf("Grade: B\n");
    } else if (marks >= 70) {
        printf("Grade: C\n");
    } else if (marks >= 60) {
        printf("Grade: D\n");
    } else {
        printf("Grade: Fail\n");
    }
    return 0;
}
```



Compiled by ab

switch case statement

switch case statement

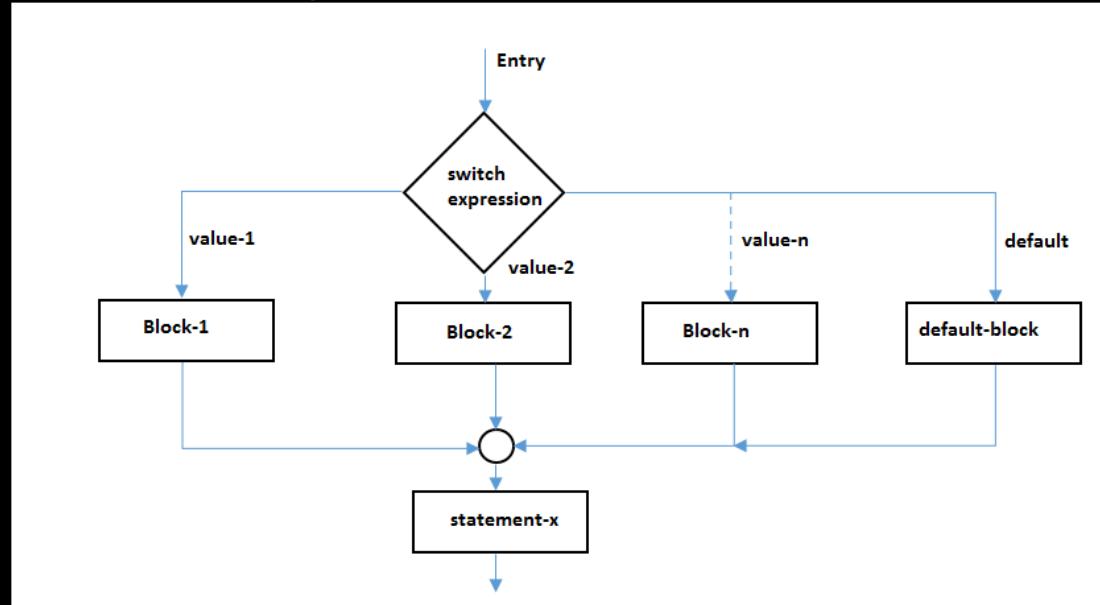


- The control statement that allows us to make a decision from number of choices is called switch case statement.
 - The switch case statement successively test the value of a given variables (an expression) with a list of case value (integer or character constants)
 - When match is found, the statement associated with that case is executed.
 - If none of the case value matches the expression then default statement is executed.

General form:

switch case statement

Compiled by ab





switch statement example:

WAP to display the corresponding days of a week according to the numbers entered.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    Compiled by ab
    int day;
    printf("Enter the numeric day of week\n");
    scanf("%d", &day);
    switch (day)
    {
        case 1:
            printf("Day is Sunday") ;
            break;
        case 2:
            printf("Day is Monday");
            break;
        case 3:
            printf("Day is Tuesday");
            break;
        case 4:
            printf("Day is Wednesday");
            break;
    }
}
```

switch statement example:

WAP to display the corresponding days of a week according to the numbers entered.

```
case 5:  
    printf("Day is Thursday");  
    break;  
  
case 6:  
    printf("Day is Friday");  
    break;  
  
case 7:  
    printf("Day is Saturday");  
    break;  
  
default:  
    printf("Invalid choice!");  
}  
  
getch();  
return 0;  
}
```



AMBITION GURU

Compiled by ab

Unit 3: Part 2

Looping Statements



Compiled by ab

Repetition/Iteration/Loop Control Instructions



Repetition/Iteration/Loop Control Instructions

Compiled by ab

Loop control instructions causes a program to execute the certain block of code until some conditions for termination of loop are satisfied.

Basically there are three types of loop control instructions.

1. while Loop (*Entry Controlled loop*)
2. do while loop (*Exit Controlled loop*)
3. for loop



Compiled by ab

Repetition/Iteration/Loop Control Instructions : **while loop**

Repetition/Iteration/Loop Control Instructions : while loop

Compiled by ab

✓ Entry Controlled loop

Test condition is evaluated first

- If the condition is true the body of the loop is executed.
- After the execution of the body, test condition is again evaluated and if it is true body is executed once again. This process goes on until test condition is false.
- If test condition is false, the body of the loop will not be executed and control is transferred out of the loop.

Repetition/Iteration/Loop Control Instructions : **while** loop

Compiled by ab

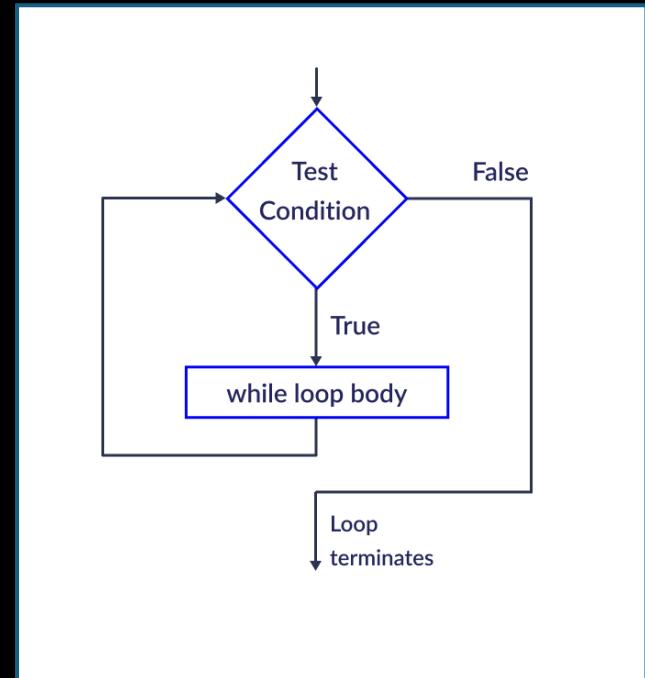
Syntax:

while (test condition)

{

body of the loop

}





while loop

**WAP to print 10 natural
numbers using while loop.**

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



Compiled by ab

Repetition/Iteration/Loop Control Instructions : **do...while loop**



Repetition/Iteration/Loop Control Instructions : do...while loop

Compiled by ab

✓ Exit Controlled loop

The test is performed at end of the body of the loop and therefore the body of the loop is executed unconditionally for the first time.

Then, test condition is evaluated.

- If test condition is true, the body of the loop is executed again. This process goes on until the test condition is false.
- When test condition is false, loop is terminated and control goes to the statement that appears immediately after the while statement.

Repetition/Iteration/Loop Control Instructions : do...while loop

Compiled by ab

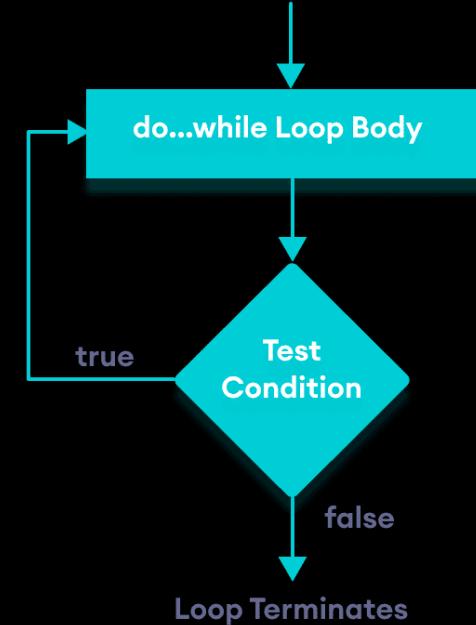
Syntax

do

{

Body of the loop

} while(test condition);





do...while loop

**WAP to print 10 natural
numbers using do...while loop.**

```
Compiling by ab
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    i=1;
    do
    {
        printf("%d\n",i);
        i++;
    }while(i<=10);
    getch();
    return 0;
}
```



Compiled by ab

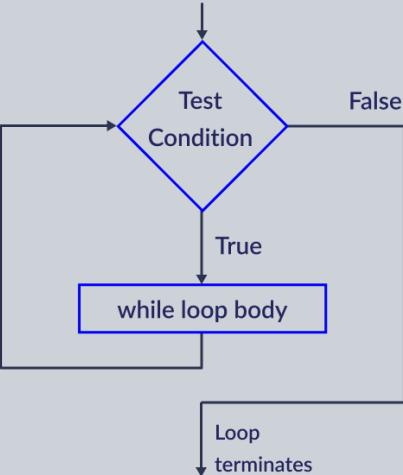
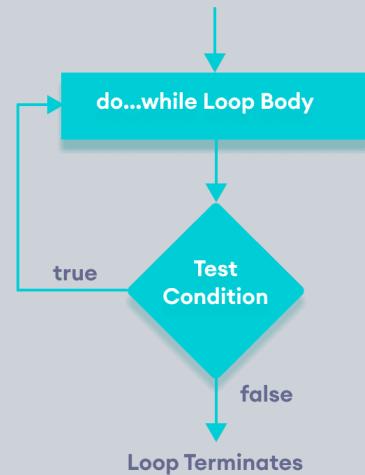
while loop Vs do...while loop



while loop(Entry controlled) Vs do...while^{loop}(Exit Controlled)

While loop	Do-while loop(Exit controlled loop)
It is entry controlled loop	It is exit controlled loop.
Test condition is evaluated at beginning of the loop execution.	Test condition is evaluated at the end of the body of the loop.
The body of the loop will execute only if the test condition is true	The body of the loop will execute at least once without depending on the test condition.
Syntax: while(test condition) { Body of the loop }	Syntax: do { Body of the loop }while(test condition);

while loop(Entry controlled) Vs do...while loop(Exit Controlled)

While loop	Do-while loop(Exit controlled loop)
 <pre>graph TD; Start(()) --> Test{Test Condition}; Test -- True --> Body[while loop body]; Body --> Test; Test -- False --> End(());</pre>	 <pre>graph TD; Body[do...while Loop Body] -- true --> Body; Body -- false --> End(());</pre>
Write an example.	Write an example.



Compiled by ab

Repetition/Iteration/Loop Control Instructions : **for loop**



Repetition/Iteration/Loop Control Instructions : for loop

Compiled by ap

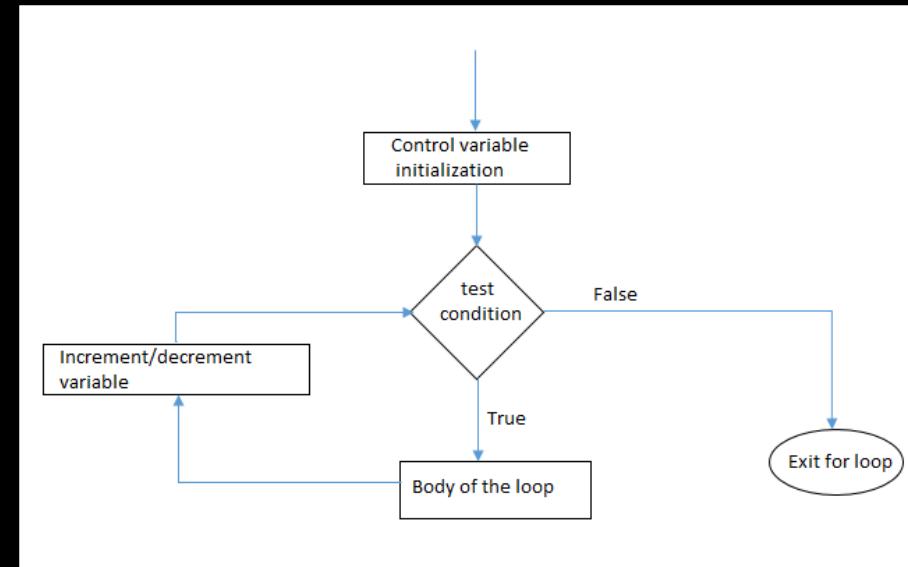
- A **for loop** in C is a control flow statement used to repeatedly execute a block of code a specific number of times. It is particularly useful when the number of iterations is known in advance.
- The syntax of a **for loop** includes initialization, a condition to check before each iteration, and an increment/decrement operation.

Repetition/Iteration/Loop Control Instructions : for loop

Compiled by ab

Syntax:

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





for loop

WAP to print 10
natural numbers using
for loop.

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

int main()
{
    int i;
    for (i=1;i<=10;i++)
    {
        printf ("%d\n", i);
    }
    getch();
    return 0;
}
```



Compiled by ab

Repetition/Iteration/Loop Control Instructions : Nested loop

Repetition/Iteration/Loop Control Instructions : Nested loop

Compiled by ab

- A loop inside another loop is called nesting of loops.
- There can be any number of loops inside one another with any combinations depending on the complexity of program.
- eg. A for loop inside a while loop or while loop inside a for loop.



Repetition/Iteration/Loop Control Instructions : Nested loop

Compiled by ab

Nested while loop

```
while(test-condition1)
{
    Statement(s);
    while(test-condition2)
    {
        statement(s);
        . . . . .
    }
    . . . . .
}
```

Nested do-while loop

```
do
{
    Statement(s);
    do
    {
        Statement(s);
        . . . . .
    }while(test-condition2);
    . . . . .
}while(test-condition1);
```



Repetition/Iteration/Loop Control Instructions : Nested loop

Compiled by ab

Nested for loop

```
for(initialization;testcondition;increment/decrement)
{
    Statement(s);
    for(initialization;testcondition;increment/decrement)
    {
        Statement(s);
    }
}
```



Nested loop

**WAP to print the
following output:**

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

Example: Nested for loop

```
#include<stdio.h>  
#include<conio.h>  
int main()  
{  
    int i,j;  
    for(i=1;i<=5;i++)  
    {  
        for(j=1;j<=i;j++)  
        {  
            printf("*\t");  
        }  
        printf("\n");  
    }  
    getch();  
    return 0;  
}
```



Compiled by ab

Unit 3: Part 3

Jump Statements



Compiled by ab

JUMP statements



JUMP statements

Compiled by ab

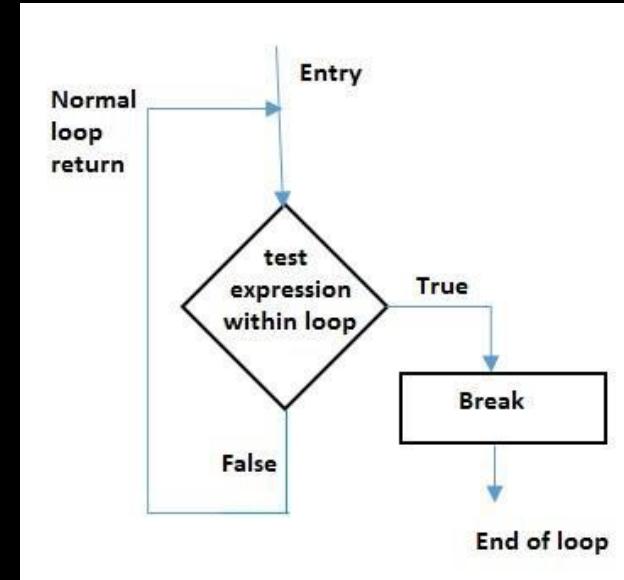
- Jump Statement makes the control jump to another section of the program unconditionally when encountered.
- It is usually used to terminate the loop or switch-case instantly. It is also used to escape the execution of a section of the program.
- There are basically three types of jump statements:
 1. Break
 2. Continue
 3. Goto

JUMP statements

Compiled by ab

i) Break statement

- The break statement is used inside the loop or switch statement.
- The execution of break statement will abort the loop and continue to execute statements followed by loop.
- Similarly, execution of break statement causes bypass the rest of statement in switch and takes control out of the switch statement.



Flowchart



Break statement example

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    for(i=1;i<=5;i++)
    {
        printf("%d\t",i);
        if(i==3)
        {
            break;
        }
    }
    printf("\nloop terminates here.");
    getch();
    return 0;
}
```

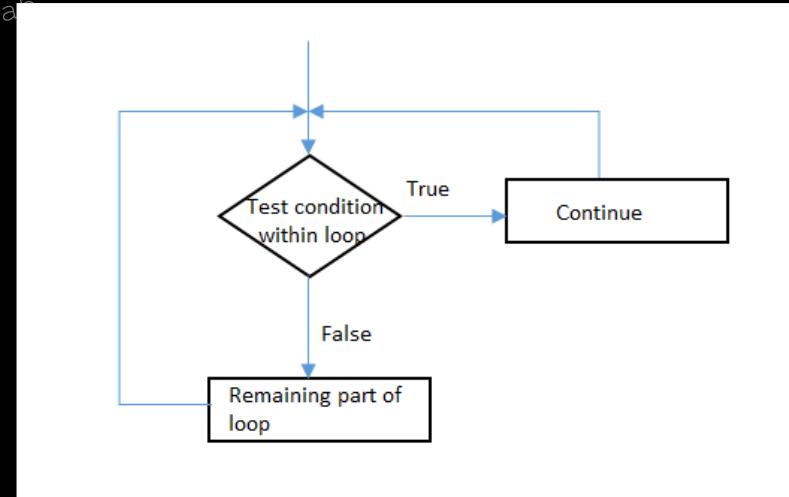
Output:
1 2 3
loop terminates here.

JUMP statements

ii) Continue statement

When continue statement is encountered inside the loop, it skips the execution of statements specified within the body of the loop and control automatically passes to the next iteration of the loop.

Compiled by a



Flowchart



```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    for(i=1;i<=5;i++)
    {
        if(i==3)
        {
            continue;
        }
        printf("%d\t",i);
    }
    getch();
    return 0;
}
```

Output:
1 2 4 5

Continue statement example

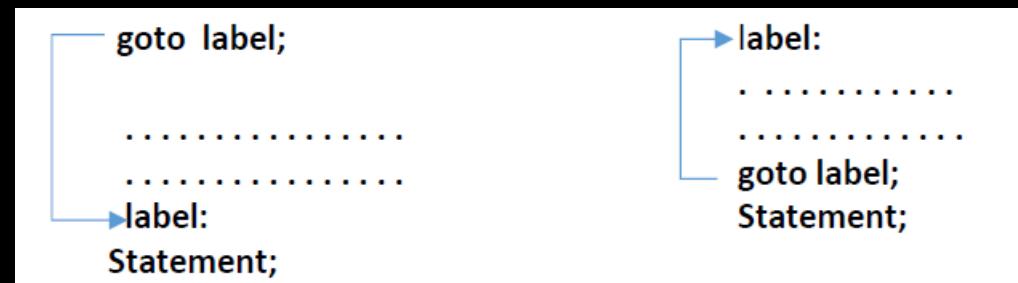


JUMP statements

iii) goto statement

The goto statement is used to alter the program execution sequence by transferring the control to some other parts of the program.

Syntax:



Forward Jump

Backward Jump

where, label is an identifier that is used to label the target statement to which control will be transferred.



```
#include<stdio.h>
#include<conio.h>
int main()
{
    int a,b;
    printf("Enter two numbers\n");
    scanf("%d%d", &a, &b);
    if(a>b)
    {
        goto labell;
    }
    else
    {
        goto label2;
    }
    labell:
    printf("Greatest number=%d", a);
    return 0;
    label2:
    printf("Greatest number=%d", b);
    getch();
    return 0;
}
```

goto statement example



Compiled by ab

Common Pitfalls: Infinite loops, Misplacement of Conditionals



Common Pitfalls: Infinite loops

An infinite loop occurs when the ^{Compiled by ab} loop condition never becomes false, causing the loop to execute indefinitely.

Missing or Incorrect Update Statement: Forgetting to update the loop control variable.

```
int i = 0;
while (i < 10) {
    // Missing i++ or similar update
    printf("%d\n", i);
}
```

Incorrect Condition: The condition may always evaluate to true.

```
int i = 0;
while (i >= 0) {
    printf("%d\n", i);
    i++;
}
```



Common Pitfalls: **Infinite loops**

Compiled by ab

Infinite Loops Prevention:

- Ensure the loop condition will eventually become false.
- Double-check the loop control variable is being updated correctly.
- Use for loops when the number of iterations is known, as they bundle initialization, condition, and update in one place.



Common Pitfalls: Misplacement of Conditionals

- Placing conditional statements (like if, else if, else) in the wrong order or incorrectly nesting them can lead to logical errors.

Common Causes:

Overlapping Conditions: Conditions that are not mutually exclusive can cause unexpected behavior.

```
int x = 10;
if (x > 5) {
    printf("x is greater than 5\n");
} else if (x > 8) {
    printf("x is greater than 8\n");
}
```

The second condition ($x > 8$) will never be evaluated because the first condition ($x > 5$) is always true for $x = 10$.

Fix: Change the logic so both conditions are checked properly.



Common Pitfalls: Off-by-One Errors

Compiled by ab

- These errors occur when a loop iterates one time too many or one time too few, often due to incorrect boundary conditions.

Common Causes:

Incorrect Loop Bounds: Using `<=` instead of `<` or vice versa

```
for (int i = 0; i <= 10; i++) {  
    printf("%d\n", i); // Prints 0 to 10 (11 iterations)  
}
```

If the intention was to print 0 to 9, the condition should be `i < 10`.



Common Pitfalls: **Off-by-One Errors**

Compiled by ab

Prevention:

- Carefully check loop conditions, especially when dealing with arrays.
- Use constants or sizeof to determine array sizes dynamically.
- Test with boundary values to ensure correct behavior.



Compiled by ab

Some Questions to Practice



Question 1

WAP to find factorial of a given number

```
#include<stdio.h>
#include<conio.h> ab
int main() compiled by
{
    int num,i,fact = 1;
    printf("Enter a number \n");
    scanf("%d", &num);
    for (i= 1; i <= num; i++)
    {
        fact = fact * i;
    }
    printf("Factorial of given number= %d \n", fact);
    getch();
    return 0;
}
```



Question 2

WAP to enter numbers
until user press zero(0)
and find the sum of
supplied numbers

```
#include<stdio.h>
#include<conio.h> ab
int main()
{
    int sum=0, num;
    do
    {
        printf("Enter number");
        scanf("%d", &num);
        sum=sum+num;
    }while(num!=0);
    printf("Total sum of Entered number=%d", sum);
    getch();
    return 0;
}
```



Question 3

WAP to check the given number is palindrome or not.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num, rem, rev=0, a;
    printf("Enter the number\n");
    scanf("%d", &num);
    a=num;
    while(num!=0)
    {
        rem=num%10;
        rev=rev*10+rem;
        num=num/10;
    }
    if(a==rev)
    {
        printf("Entered number is palindrome");
    }
    else
    {
        printf("Entered number is not palindrome");
    }
    getch();
    return 0;
}
```



Question 4

WAP to Check the given
number is Armstrong
number or not

```
#include <stdio.h>
#include<conio.h>
#include <math.h>
int main()
{
    int num,orgnum,rem,sum=0,n=0;
    printf("Enter the number: ");
    scanf("%d", &orgnum);
    num=orgnum;
    while( num != 0 )
    {
        num =num/10;
        ++n;
    }
    num=orgnum;
```



Question 4

WAP to Check the given
number is Armstrong
number or not

```
while (num != 0)
{
    rem = num%10;
    sum = sum+pow(rem, n);
    num=num/10;
}
if(sum == orgnum)
{
    printf("Given number is armstrong number");
}
else
{
    printf("Given number is not an Armstrong
number");
}
getch();
return 0;
}
```



Question 5

WAP to find whether the entered year is leap year or not.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    Compiled by ab
    int year;
    printf("Enter the year\n");
    scanf("%d", &year);
    if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))
    {
        printf("%d is a leap year", year);
    }
    else
    {
        printf("%d is not a leap year", year);
    }
    getch();
    return 0;
}
```



Question 6

Write a program and draw a flowchart to read a positive integer value and compute the following sequence .If the number is even, half it ,if it is odd ,multiply 3 and add 1 print the result. If the input value is less than 1, print a message containing word “ERROR”.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int num;Compiled by ab
    printf("Enter the integer number\n");
    scanf("%d", &num);
    if(num<1)
    {
        printf("ERROR");
    }
    else
    {
        if (num%2==0)
        {
            num=num/2;
        }
        else
        {
            num=num*3+1;
        }
        printf("Number after result=%d", num);
    }
    getch();
    return 0;
}
```



AMBITION GURU

Compiled by ab

Questions



Questions

Compiled by ab

1. How does sequential control structure differ from decision control structure.
2. Write a C program that reads the age of a person and checks if they are eligible to get citizenship. A person is eligible to get citizenship if they are 16 years or older. If they are eligible, display a message indicating that they can get citizenship; otherwise, display a message indicating that they are not eligible.
3. Write syntax and draw flowchart : if statement, if...else statement, nested if else statement, else if ladder, switch statement
4. Write a C program to input an integer and check whether it is even or odd using if-else.
5. WAP to display the corresponding month of a year according to the numbers entered. (1 – Jan, 2- Feb, etc.)
6. WAP to input three numbers and find the largest number using nested if else statement.
7. WAP to input a number and check whether it is positive, negative or zero.
8. Make a comparison between nested if else statement and else if ladder statement with their syntax, flowchart and example for each.



Questions

- Compiled by ab*
9. Write a program and draw a flowchart to read a positive integer value and compute the following sequence .If the number is even, half it ,if it is odd ,multiply 3 and add 1 print the result. If the input value is less than 1, print a message containing word “ERROR”.
 10. WAP to Check the given number is Armstrong number or not.
 11. WAP to check the given number is palindrome or not.
 12. WAP to enter numbers until user press zero(0) and find the sum of supplied numbers
 13. Write syntax and draw flowchart : while statement, for statement, do...while statement
 14. Differentiate between entry-controlled loop and exit controlled loop.
 15. Explain in brief about continue, go to and break statement with syntax and example for each.
 16. Explain common pitfalls for control structures in brief .



AMBITION GURU

Compiled by ab

THANK YOU
Any Queries ?