Programming for problem solving using C

123ES

Ms. Urvashi Gohil
Assistant Professor

Birla Vishvakarma Mahavidhyalaya (BVM) Engineering College
Department of Computer Engineering
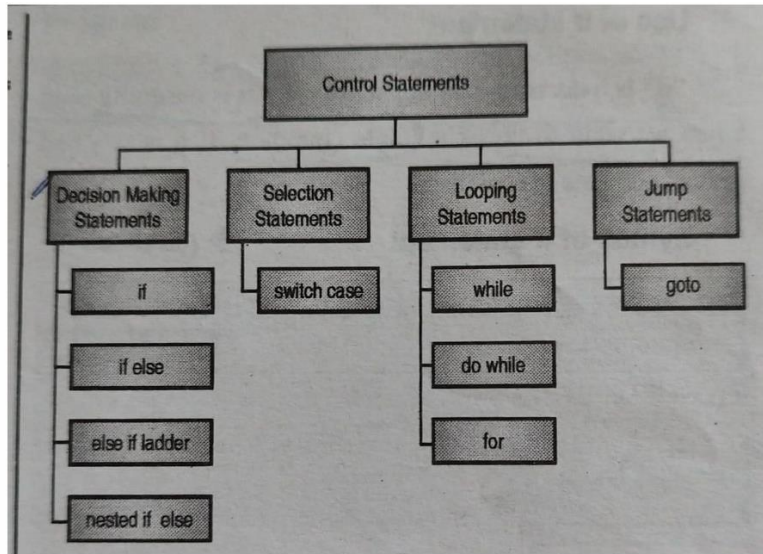
January 31, 2025

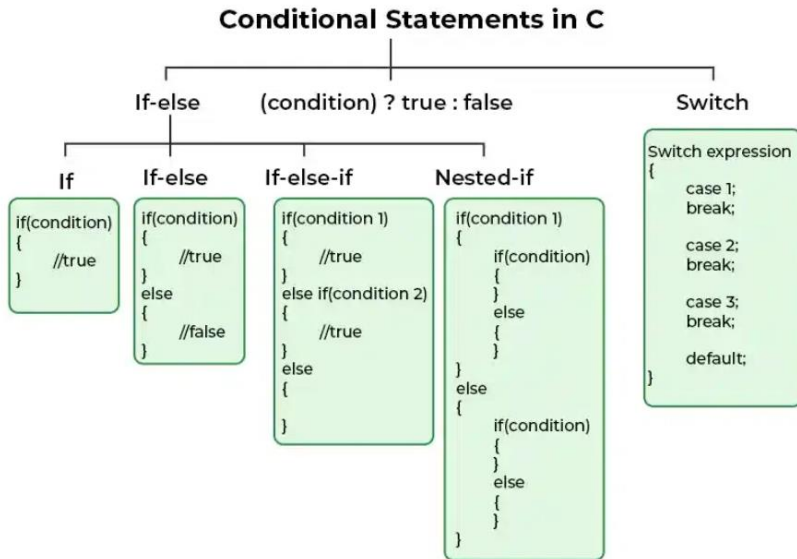Outline of the course :

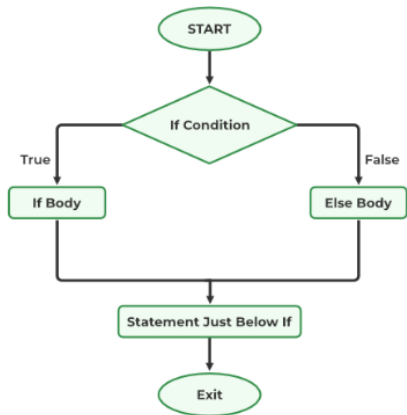Conditional branching

Unconditional branching

Looping

break and continue

Conditional branching

Conditional Statements in C
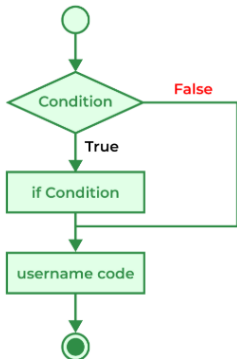
Task

Write a program which does the following
- Take input for two integer variables `a` & `b`
- Output "Coding is Fun" to the console if a is greater than b.

Sample 1:

Input		Output	
25 20		Coding is Fun	

Sample 2:

Input		Output	
20 20			

```c
1   #include <stdio.h>
2
3   int main() {
4       int a;
5       int b;
6
7       scanf("%d", &a);
8       scanf("%d", &b);
9
10      if (a > b) {
11          printf("Coding is Fun");
12      }
13  }
```

Sample Input

```
25
20
```

Your Output

```
Coding is Fun
```

```c
abc > C getc.c > ⊘ main()
1    #include<stdio.h>
2    int main(){
3       //if(); -->; makes sentence empty and print next two statements
4
5       if(0); // means what if condition is false-->0?
6
7       printf("its true");
8
9       printf("\nits wrong");
10
11   }
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'getc.exe'
its true
its wrong
PS F:\code1\abc\output> 
```

Task

Write a program which does the following:
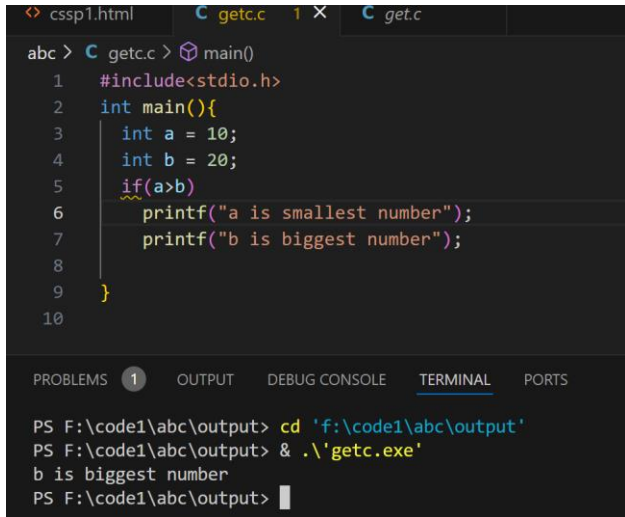
- Let's think of a real-life example where we need to find out if a person is old enough to vote.
- Find out if the age (25) is greater than OR equal to the voting age limit, which is set to 18.
- Declare the variables `age` and `voting_age` - and initialize them to the `values` `25` and `18` - i.e. the age and the voting age respectively.
- Compare `age` and `voting_age` using the syntax given above and output the following
  - `"Old enough to vote!"` if `age` is greater than or equal to `voting_age`
  - `"Not old enough to vote."` if `age` is lesser than `voting_age`

```c
#include <stdio.h>

int main() {
    int age = 25;
    int voting_age = 18;

    if (age >= voting_age) {
        printf("Old enough to vote!");
    } else {
        printf("Not old enough to vote.");
    }
}
```

Your Output

```
Old enough to vote!
```

```c
#include<stdio.h>
int main(){
  int a = 10;
  int b = 20;
  if(a>b)
    printf("a is smallest number");
    printf("b is biggest number");

}
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'getc.exe'
b is biggest number
PS F:\code1\abc\output>
```

If condition without {curly braces/ body} will execute first statement if condition true

If condition false, it will execute next statement.

If else with printf scanf

```c
#include <stdio.h>

int main() {
    int a;
    if (printf("Hello, World!\n") && scanf("%d", &a)) {  // printf returns a nonzero value (number of characters p
        printf("Condition is TRUE\n");                    //Scanf also returns int(non zero value)
    } else {
        printf("Condition is FALSE\n");
    }
    return 0;
}
```

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'ifprin.exe'
Hello, World!
3
Condition is TRUE
PS F:\code1\abc\output>
```

If Else If Statement

if else if ladder is an extension of if else statement used to test a series of conditions sequentially, executing the code for the first true condition. A condition is checked only if all previous ones are false. Once a condition is true, its code block executes, and the ladder ends"
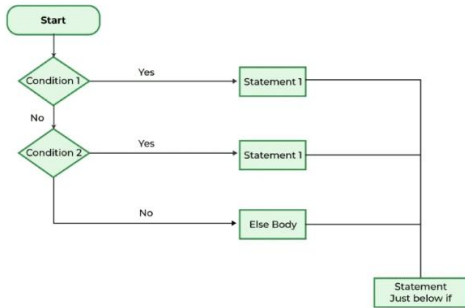
```c
abc > C elif.c > ⊘ main()
1   #include<stdio.h>
2   int main(){
3   int grade = 85;
4   if (grade >= 90) {
5       printf("You got an A");
6   } else if (grade >= 80) {
7       printf("You got a B");
8   } else if (grade >= 70) {
9       printf("You got a C");
10  } else {
11      printf("You need to study more");
12  }
13  }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'elif.exe'
You got a B
PS F:\code1\abc\output>
```

Task

Write a program which does the following

- Take two integers `b` and `r` as input
- Print *"Rob scored higher marks than Bob"*, if `r` is greater than `b`
- Print *"Bob & Rob both scored the same"*, if both `b` and `r` are equal

Sample 1:

Input	Output
20 25	Rob scored higher marks than Bob

Sample 2:

Input	Output
15 15	Bob & Rob both scored the same

```c
1   #include <stdio.h>
2
3   int main() {
4       int b, r;
5       scanf("%d", &b);
6       scanf("%d", &r);
7       if(r>b){
8           printf("Rob scored higher marks than Bob");
9       }
10      else if(b == r){
11          printf("Bob & Rob both scored the same");
12      }
13
14  }
15
```

Sample Input

```
20
25
```

Your Output

```
Rob scored higher marks than Bob
```
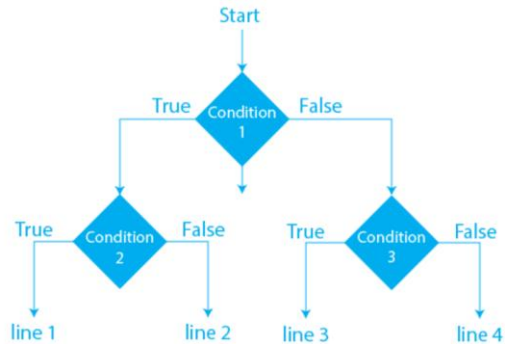
```c
abc > C ifels.c > ⊕ main()
1    #include <stdio.h>
2    int main()
3    {
4        int i = 10;
5        if (i == 10) {
6            // First if statement
7            if (i < 15)
8                printf("i is smaller than 15\n");
9
10           if (i < 12)
11               printf("i is smaller than 12 too\n");
12           else
13               printf("i is greater than 15");
14       }
15       else {
16           if (i == 20) {
17
18               if (i < 22)
19                   printf("i is smaller than 22 too\n");
20               else
21                   printf("i is greater than 25");
22           }
23       }
24   }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
i is smaller than 15
i is smaller than 12 too
```

Atm money Withrow

Step-1 check the correct pin (pin_entered = correct_pin)
Step-2 check the amount > 0 or not? If not then invalid amount
Step-3 check the amount <= balance (Sufficient amount)
      if yes→ Successful transition
      if not→ not enough balance

Q: If the number is even or odd, and then if it is even, whether it is divisible by 4 or not, and if it is odd, whether it is divisible by 3 or not

```c
#include<stdio.h>
int main(){
    int enter_pin, correct_pin, amount, balance;
    correct_pin = 1234;
    balance = 100;
    printf("Enter the pin number: ");
    scanf("%d", &enter_pin);
    if(enter_pin == correct_pin)
    {
        printf("Enter the amount: ");
        scanf("%d", &amount);
        if(amount > 0)
        {
            if(amount <= balance)
            {
                printf("Successful transaction\n");
            }
            else {
                printf("Insufficient balance\n");
            }
        }
        else {
            printf("Enter a valid amount\n");
        }
    }
    else {
        printf("Enter a valid pin number\n");
    }
}
```

```c
#include<stdio.h>
int main(){
    int n;
    printf("Enter the number:");
    scanf("%d", &n);

    if(n%2 == 0){
        printf("Even number");

        if(n %4 == 0){
            printf("\nnumber is div by 4");
        } else printf("\nnumber is not div by 4");
    }
    else{
        printf("Odd number");
        if (n % 3 == 0){
            printf("\nnumber is div by 3");
        }
        else printf("\nnumber is not div by 3");
    }
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'odev.exe'
Enter the number:3
Odd number
number is div by 3
```

3	*,/,%	Multiplication, division, modulus	Left-to-Right
4	+/-	Addition, subtraction	Left-to-Right
5	<< , >>	Bitwise shift left, Bitwise shift right	Left-to-Right
6	< , <=	Relational less than, less than or equal to	Left-to-Right
	> , >=	Relational greater than, greater than or equal to	
7	== , !=	Relational is equal to, is not equal to	Left-to-Right

```
abc > C cpe1.c > main()
  1    #include<stdio.h>
  2    int main(){
  3        int x = 5;
  4        if(0 <= x <=4){
  5            printf("condition is true!");
  6        }
  7        else printf("condition is false");
  8    }
```

```
PROBLEMS  2    OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'cpe1.exe'
condition is true!
PS F:\code1\abc\output>
```

Left to right associativity
For any positive value of
x, condition will be true,

0<=x→1
1 <=4 →1

```
abc > C cpe1.c > ⊗ main()
1    #include<stdio.h>
2 ∨  int main(){
3       int x = 15;
4 ∨     if(x = 10)
5       {
6        printf("x is 10");
7       }
8       else printf("x is 15");
9    }
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'cpe1.exe'
x is 10
PS F:\code1\abc\output>
```

= is assignment operator
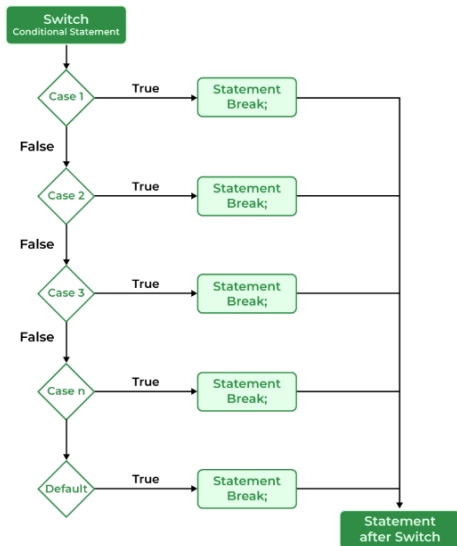
Permitted datatype for switch:
Int, ch, short
Not permitted: float and double

```c
abc > C p-3.c > ⊙ main()
1    #include<stdio.h>
2  ∨ int main(){
3        int grade = 79;
4        int score = grade/10;
5
6  ∨     switch (score)
7        {
8        case 9:
9            printf("You Got A");
10           break;
11       case 8:
12           printf("you got B");
13           break;
14       case 7: |
15           printf("you got C");
16           break;
17       default:
18           printf("you got D");
19           break;
20       }
21   }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> & .\'p-3.exe'
you got C
PS F:\code1\abc\output> 
```

Switch case with Enum

```c
abc > C enum.c > main()
  1   #include<stdio.h>
  2   enum Week {
  3       sunday, monday, tuesday, wed, thursday, friday, saturday
  4   }w;
  5   int main(){
  6       printf("Enter the day: ");
  7       scanf("%d", &w);
  8       switch (w)
  9       {
 10       case sunday:
 11           printf("Holiday");
 12           break;
 13       case monday:
 14           printf("Working Day");
 15           break;
 16       case tuesday:
 17           printf("Tuesday");
 18           break;
 19       case wed:
 20           printf("early free");
 21           break;
 22       case thursday:
 23           printf("thursday");
 24           break;
 25       case friday:
 26           printf("friday");
 27           break;
 28       case saturday:
 29           printf("saturday");
```

```
PROBLEMS 1    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'enum.exe'
Enter the day: 3
early free
PS F:\code1\abc\output>
```

Enum

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign **names to integral constants**, the names make a program easy to read and maintain.

```c
abc > C en.c
1    #include<stdio.h>
2  v enum Os_states {
3        new = 1, ready, run, block, terminate
4    }state;
5
6  v int main(){
7        state = run;
8        printf("%d", state); // op is 3 if default (new = starting from 0) op is 2
9    }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'en.exe'
3
PS F:\code1\abc\output>
```

Facts about enum:

1. Two enum names can have same value.
2. If we do not explicitly assign values to enum names, the compiler by default assigns values starting from 0.
3. We can assign values to some name in any order. All unassigned names get value as value of previous name plus one.
4. The value assigned to enum names must be some integral constant, i.e., the value must be in range from minimum possible integer value to maximum possible integer value.
5. All enum constants must be unique in their scope.

Example:
1. Traffic signal system with enum and switch
2. Create a ticket booking system with enum and switch

```c
abc > C en.c > ⊗ main()
1    #include<stdio.h>
2    enum traffic_signals{
3        yellow =1 , red, green
4    }ts;
5    int main(){
6        printf("Enter 1 for yello , 2 for red, 3 for green :");
7        scanf("%d", &ts);
8        switch (ts)
9        {
10       case 1:
11           printf("Ready to go");
12           break;
13       case 2:
14           printf("Stop");
15           break;
16       case 3:
17           printf("Go");
18           break;
19       default:
20           printf("Study the traffic rules!");
21           break;
22       }
```

```
PROBLEMS  1    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'en.exe'
Enter 1 for yello , 2 for red, 3 for green :1
Ready to go
PS F:\code1\abc\output>
```

```c
#include <stdio.h>
enum traffic_signals {
    red,
    yellow,
    green
}tf;
int main() {
    int tf;
    printf("Enter the traffic signal (0 for red, 1 for yellow, 2 for green): ");
    scanf("%d", &tf);
    enum traffic_signals ts = (enum traffic_signals)tf;
    switch (ts) {
        case red:
            printf("Stop! Red light.\n");
            break;
        case yellow:
            printf("Caution! Yellow light.\n");
            break;
        case green:
            printf("Go! Green light.\n");
            break;
        default:
```

```
Go! Green light.
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'q-1.exe'
Enter the traffic signal (0 for red, 1 for yellow, 2 for green): 1
Caution! Yellow light.
PS F:\code1\abc\output>
```

Switch case with if-else

```c
abc > C switchifelse.c > ⊗ main()
1   #include <stdio.h>
2   int main() {
3       int choice, beverageType;
4       printf("Restaurant Menu:\n");
5       printf("1. Burger\n");
6       printf("2. Beverage\n");
7       printf("Enter your choice (1-3): ");
8       scanf("%d", &choice);
9       switch (choice) {
10          case 1:
11              printf("You ordered a Burger. Preparing your order...\n");
12              break;
13          case 2:
14              printf("Choose Beverage Type:\n");
15              printf("1. Hot Coffee\n");
16              printf("2. Cold Juice\n");
17              printf("Enter your choice (1-2): ");
18              scanf("%d", &beverageType);
19
20              if (beverageType == 1) {
21                  printf("Serving a Hot Coffee.\n");
22              } else if (beverageType == 2) {
23                  printf("Serving a Cold Juice.\n");
24              } else {
25                  printf("Invalid choice! Serving water instead.\n");
26              }
27              break;
28          default:
29              printf("Invalid selection! Please choose a valid menu item.\n");
30      }
31  }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Restaurant ordering system:
- Two variables category & food choice.
- Category stores two item fast food and beverages. (Use switch case for this 2)
- Food choice: pizza , burger (use switch case for this choice)
- Beverage's choice: coffee, cold drinks(use switch case for this)

```c
#include <stdio.h>
int main() {
    int category, foodChoice;
    // Displaying the main menu categories
    printf("Welcome to the Restaurant!\n");
    printf("Choose a category:\n");
    printf("1. Fast Food\n");
    printf("2. Beverages\n");
    printf("Enter your choice (1-2): ");
    scanf("%d", &category);
    switch (category) {
        case 1: // Fast Food category
            printf("Fast Food Menu:\n");
            printf("1. Burger\n");
            printf("2. Pizza\n");
            printf("Enter your choice (1-2): ");
            scanf("%d", &foodChoice);
            switch (foodChoice) {
                case 1:
                    printf("You ordered a Burger. Preparing your order...\n");
                    break;
                case 2:
                    printf("You ordered a Pizza. Preparing your order...\n");
                    break;
                default:
                    printf("Invalid choice in Fast Food category.\n");
            }
            break;
        case 2: // Beverages category
            printf("Beverage Menu:\n");
            printf("1. Hot Coffee\n");
            printf("2. Cold Juice\n");
            printf("Enter your choice (1-2): ");
            scanf("%d", &foodChoice);
            switch (foodChoice) {
                case 1:
                    printf("Serving a Hot Coffee.\n");
                    break;
                case 2:
                    printf("Serving a Cold Juice.\n");
                    break;
                default:
                    printf("Invalid choice in Beverages category.\n");
            }
            break;
        default:
            printf("Invalid category selection!\n");
    }
}
```

As switch does not support float, this example show how to map float values in supportive data type(int, ch)

```c
#include <stdio.h>

int main() {
    float cgpa;
    int gradeCategory; // Integer mapping for CGPA

    printf("Enter cgpa:");
    scanf("%f", &cgpa);

    if(cgpa >= 8.0 && cgpa <=100.0)
    {
        gradeCategory = 1;
    }
    else if (cgpa >= 7.9 && cgpa <= 7.0)
    {
        gradeCategory = 2;
    }

    printf("To check your award please enter your gradecategory");
    scanf("%d",&gradeCategory);


    switch (gradeCategory)
    {
    case 1:
      printf("You won gold medal");
      break;

    case 2:
      printf("You won silver medal");
      break;
    default:
      printf("you won chocolet box");
      break;
    }
}
```

- jump to some part of the code

The goto statement allows transfer control of the program to the specified label

```c
abc > C goto.c > ⓧ main()
1    #include <stdio.h>
2
3    int main() {
4        int n = 24;
5
6        // If the number is even, jump to
7         // jump_here label
8        if (n %2 == 0)
9            goto jump_here;
10
11        // This will be skipped
12        printf("odd number \n", n);
13
14   jump_here:
15        printf("Even number\n");
16        return 0;
17   }
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Even number
PS F:\code1\abc\output>

Goto Statement

```
PS F:\code1\abc\output> & .\'gotojump.exe'
Outer loop executing. i = 0
 Inner loop executing. j = 0
 Inner loop executing. j = 1
Outer loop executing. i = 1
 Inner loop executing. j = 0
Jumped to stop. i = 1
PS F:\code1\abc\output>
```

```c
abc > C gotojump.c > ⬡ main()
 1    #include <stdio.h>
 2    int main()
 3    { int i,j;
 4        for (i = 0; i < 6; i++ )
 5        {
 6            printf( "Outer loop executing. i = %d\n", i );
 7            for (j = 0; j < 2; j++ )
 8            {
 9                printf( " Inner loop executing. j = %d\n", j );
10                if ( i == 1 )
11                    goto stop;
12            }
13        }
14        /* This message does not print: */
15        printf( "Loop exited. i = %d\n", i );
16        stop: printf( "Jumped to stop. i = %d\n", i );
17    }
```

An unrestricted use of the "goto" statement is harmful because
(a) it makes it more difficult to verify programs
(b) it increases the running time of the programs
(c) it increases the memory required for the programs
(d) it results in the compiler generating longer machine code

While loop

While (condition)
        statement_to_repeat

While(Condition){
        statement 1;
        .
        .
        Statement N;

        }

Counter controlled loop & sentinel control loop

```c
#include<stdio.h>
int main(){
    int count;
    printf("Enter the count:");
    scanf("%d",&count);

    printf("countdown starts!");

    while (count >= 0)
    {
        sleep(1);
        printf("\n%d", count);
        count --;
    }

    printf("\nTimes up");
}
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'whil.exe'
Enter the count:3
countdown starts!
3
2
1
0
Times up
PS F:\code1\abc\output>
```

```c
abc > C factwhil.c > ⊗ main()
 1    #include<stdio.h>
 2    int main(){
 3        int fact = 1 , n;
 4        printf("Enter the number:");
 5        scanf("%d", &n);
 6        while(n>1){
 7            fact  *=  n ;   //fact = fact * n;
 8            n--;
 9        }
10        printf("\nThe fact is:  %d ",fact);
11    }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'factwhil.exe'
Enter the number:8

The fact is:  40320
PS F:\code1\abc\output> 
```

```c
abc > C w.c > ⊘ main()
1    #include<stdio.h>
2    int main(){
3     int i = 1;
4     while(i ==1){
5        printf("%d", i);
6        i++;
7     }
8     printf("\nThe value of i is %d", i);
9    }
10
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'w.exe'
1
The value of i is 2
PS F:\code1\abc\output> []
```

```
abc > C w.c > ⊗ main()
1    #include<stdio.h>
2    int main(){
3     int i = 1;
4     while(i =1){
5        printf("%d", i);
6        i++;
7     }
8     printf("\nThe value of i is %d", i);
9    }
10
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111
```

```c
abc > C w.c > ⊙ main()
1    #include<stdio.h>
2    int main(){
3      int i = 1;
4      while(1){
5        printf("%d", i);
6        i++;
7      }
8      printf("\nThe value of i is %d", i);
9    }
10
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
30234730334730434730534730634730734730834730934731034731134731234731 3
31834731934732034732134732234732334732434732534732634732734732834732 9
33434733534733634733734733834733934734034734134734234734334734434734 5
35034735134735234735334735434735534735634735734735834735934736034736 1
36634736734736834736934737034737134737234737334737434737534737634737 7
38234738334738434738534738634738734738834738934739034739134739234739 3
39834739934740034740134740234740334740434740534740634740734740834740 9
41434741534741634741734741834741934742034742134742234742334742434742 5
43034743134743234743334743434743534743634743734743834743934744034744 1
```

1 is considered as true
Control comes to body
I printed and incremented
→ Infinite loop

While loop

```
abc >  C w.c >  main()
   1    #include<stdio.h>
   2    int main(){
   3     int i = -1;
   4     while(i){
   5       printf("%d", i);
   6       i++;
   7     }
   8     printf("\nThe value of i is %d", i);
   9    }
  10
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'w.exe'
-1
The value of i is 0
PS F:\code1\abc\output>
```

Not an infinite loop,

Not every time I will be incremented but it will stop at 0

It remove i++ it will infinite loop

```
abc > C w.c > ⊙ main()
  1    #include<stdio.h>
  2
  3    int main(){
  4        int i = -1;
  5        while(i){
  6            printf("The value of i is %d", i);
  7        }
  8    }
  9
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
value of i is -1The value of i is -1The value of i is -1The value of i is -1The valu
The value of i is -1The value of i is -1The value of i is -1The value of i is -1The
s -1The value of i is -1The value of i is -1The value of i is -1The value of i is -1
 i is -1The value of i is -1The value of i is -1The value of i is -1The value of i i
e of i is -1The value of i is -1The value of i is -1The value of i is -1The value of
value of i is -1The value of i is -1The value of i is -1The value of i is -1The valu
The value of i is -1The value of i is -1The value of i is -1The value of i is -1The
s -1The value of i is -1The value of i is -1The value of i is -1The value of i is -1
 i is -1The value of i is -1The value of i is -1The value of i is -1The value of i i
```

An infinite loop,

Value is not incremented so it considered as n true

```c
abc > C w.c > ⊕ main()
1    #include<stdio.h>
2    int main(){
3     int i ;
4     while(i<10){
5       printf("%d", i);
6       i++;
7     }
8     printf("\nThe value of i is %d", i);
9    }
10
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'w.exe'

The value of i is 2543616
PS F:\code1\abc\output>
```

Without init I, it gives garbage value

Loop is not executed

```c
abc > C whi.c > ⊘ main()
1    #include <stdio.h>
2
3    int main() {
4        int num, sum = 0;
5        printf("Enter positive numbers to sum (enter a negative number to stop):\n");
6        while (1) {  // Infinite loop
7            printf("Enter a number: ");
8            scanf("%d", &num);
9            if (num < 0) {
10               break;  // Exit the loop if the number is negative
11           }
12           sum += num;  // Add the number to the sum
13       }
14       printf("Total sum: %d\n", sum);
15   }
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: 4
Enter a number: -9
Total sum: 52
```

While loop with if and break statement

```c
#include<stdio.h>
int main(){
    int i=0;
    while(1){
        // do your work.
        printf("\n%d", i);
        if ( i == 10 ) break;
        i++;
    }
    printf("\nAfter While");
}
```

```c
#include<stdio.h>
int main(){
    int i = 0;
    while(i < 10){
        printf("\nhello");
        i++;
        if(i ==3 )break;
    }
    printf("\nExit of the loop ");
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

0
1
2
3
4
5
6
7
8
9
10
After While
PS F:\code1\abc\output>
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'break.exe'

hello
hello
hello
Exit of the loop
PS F:\code1\abc\output>
```

```
abc > C wh.c > ...
13
14  ∨ int main() {
15        int x = 5;
16
17        while (x-- > 0)
18  ∨         while (x % 2 == 0) {   // Inner loop runs only when x is even
19                printf("%d ", x);
20                break;
21            }
22
23        return 0;
24    }
25
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'wh.exe'
4 2 0
PS F:\code1\abc\output>
```

```
27    int main() {
28        int num = 5;
29
30        while (printf("%d ", num--) && num > 0) {}  // Prints and decrements
31
32        return 0;
33    }
34
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'wh.exe'
4 2 0
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'wh.exe'
5 4 3 2 1
PS F:\code1\abc\output>
```

```
35
36  int main() {
37      int num;
38
39      printf("Enter numbers (0 to stop):\n");
40
41      while (scanf("%d", &num) && num != 0) {
42          printf("You entered: %d\n", num);
43      }
44
45      printf("Loop ended.\n");
46      return 0;
47  }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'wh.exe'
Enter numbers (0 to stop):
6
You entered: 6
4
You entered: 4
7
You entered: 7
1
You entered: 1
0
Loop ended.
PS F:\code1\abc\output>
```

Ch in while statement

```c
#include<stdio.h>
//
int main(){
  char ch = 'b';
  while(ch > 0 ) {
    printf("%d ", ch);
    ch++;
  }
}
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'w.exe'
98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 12
2 123 124 125 126 127
PS F:\code1\abc\output>
```

Loop execute until it found
sentinel value(SENTIVAL)

Step-1 Get the line of data
Step-2 while sentival not found:
3. Process the data
4. Get another line of data.

```c
abc > C w.c > ⊗ main()
1    #include<stdio.h>
2    #define SENTIVAL -99
3    int main(){
4      int score , sum = 0;
5      printf("Enter the score:");
6      scanf("%d", &score);
7      while(score!=SENTIVAL){
8        sum += score;
9        printf("Enter the next score: ", SENTIVAL);
10       scanf("%d", &score);
11     }
12     printf("The sum of the score is %d", sum);
13   }
```

```
PROBLEMS 1    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> & .\'w.exe'
Enter the score:39
Enter the next score: 34
Enter the next score: 38
Enter the next score: -99
The sum of the score is 111
PS F:\code1\abc\output>
```
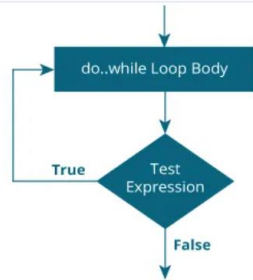
Do while statement

The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.

The syntax of the do...while loop is

do {
 // the body of the loop
}
while (Condition/testexpression);

→body of loop is executed once→check the condition
→if true then again executed the loop and then again
Check the condition→ continue till true→if false , loop ends.

Do while vs While statement

```c
abc > C c.c > ⊘ main()
1    #include<stdio.h>
2    int main(){
3        int i = 0;
4        do{
5            printf("\nhello");
6        }
7        while(i>0);
8        printf("\nout of dowhile");
9    }
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'c.exe'

hello
out of dowhile
PS F:\code1\abc\output>
```

```c
abc > C c.c > ⊘ main()
1    #include<stdio.h>
2    int main(){
3        int i = 0;
4        while(i>0){
5            printf("\nhello");
6        }
7        printf("\nout of dowhile");
8    }
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'c.exe'

out of dowhile
PS F:\code1\abc\output>
```

```c
abc > C dem1.c > ⊙ main()
1   // Program to add numbers until the user enters zero
2
3   #include <stdio.h>
4   int main() {
5     int number, sum = 0;
6
7     do {
8         printf("Enter a number: ");
9         scanf("%d", &number);
10        sum += number;
11    }
12    while(number != 0);
13
14    printf("Sum = %d",sum);
15
16  }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'dem1.exe'
Enter a number: 3
Enter a number: 4
Enter a number: 5
Enter a number: 6
Enter a number: 0
Sum = 18
PS F:\code1\abc\output>
```

Do while statement– cannot put ; after do

```
abc > C w.c > ⊗ main()
 1    #include<stdio.h>
 2
 3    int main(){
 4      int a=1;
 5        do;
 6        {
 7          printf("%d",a);
 8          a++;
 9        }
10        while(a<=5);
11    }
12
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
f:\code1\abc\w.c:5:9: warning: suggest braces around empty body in 'do'
    5 |        do;
      |          ^
f:\code1\abc\w.c:6:7: error: expected 'while' before '{' token
    6 |        {
      |        ^
```

Do while statement

```c
abc > C d.c > ⊙ main()
1    #include <stdio.h>
2
3    int main() {
4        int num, count = 0;
5        printf("Enter a number: ");
6        scanf("%d", &num);
7        if (num == 0)  count = 1;
8        do {
9            num /= 10;  // Remove the last digit
10           count++;  // Increase the count
11       } while (num != 0);  // Continue until all digits are removed
12
13       printf("Number of digits: %d\n", count);
14
15   }
16
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'d.exe'
Enter a number: 389
Number of digits: 3
PS F:\code1\abc\output>
```

```c
abc > C dem1.c > ⊗ main()
 2
 3    int main() {
 4        int n, a = 0, b = 1, next, i = 0;
 5
 6        // Taking user input
 7        printf("Enter the number of terms: ");
 8        scanf("%d", &n);
 9
10        // Printing the first Fibonacci number
11        printf("Fibonacci Series: %d, %d", a, b);
12
13        // Using do-while loop for Fibonacci sequence
14        do {
15            next = a + b;
16            printf(", %d", next);
17            a = b;
18            b = next;
19            i++;
20        } while (i < n - 2); // Because first two numbers are already printed
21    }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13
PS F:\code1\abc\output> []
```
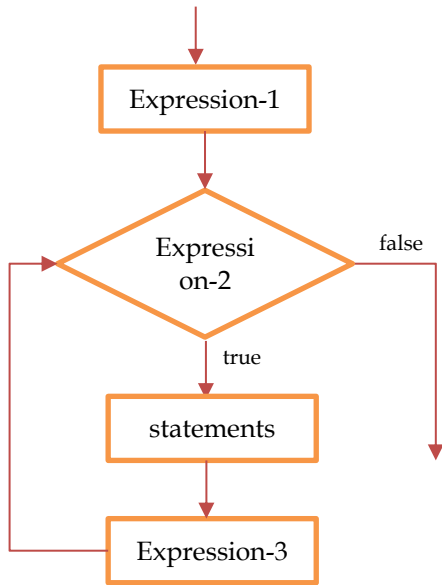
For Loop

In a for loop there are 3 expressions

expression-1 is used to initialize some variable (called index) that controls the loop.

Expression-2 is represent the condition that must be true for the loop to continue

Third expression used to alter the value of index initially assigned by expression-1

```
abc > C just.c > ⊕ main()
  1   #include<stdio.h>
  2   int main(){
  3       int i;
  4       for(i = 1; i<=5;i++){
  5           printf("\ni = %d", i);
  6       }
  7       printf("\nThe value of i after for loop is %d", i);
  8   }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

i = 1
i = 2
i = 3
i = 4
i = 5
The value of i after for loop is 6
PS F:\code1\abc\output>
```
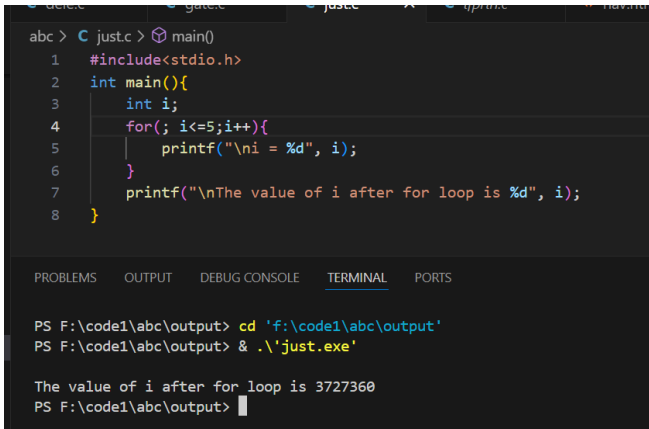
All expression are optional

Case:1 without initialization


```c
#include<stdio.h>
int main(){
    int i;
    for(; i<=5;i++){
        printf("\ni = %d", i);
    }
    printf("\nThe value of i after for loop is %d", i);
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

The value of i after for loop is 3727360
PS F:\code1\abc\output>
```

For Loop Properties

All expression are optional
-- experession-1--

Case:1 without initialization

Case:2 Initialization before for loop


```c
#include<stdio.h>
int main(){
    int i;
    for(; i<=5;i++){
        printf("\ni = %d", i);
    }
    printf("\nThe value of i after for loop is %d", i);
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

The value of i after for loop is 3727360
PS F:\code1\abc\output>
```



```c
#include<stdio.h>
int main(){
    int i=1;
    for(; i<=5;i++){
        printf("\ni = %d", i);
    }
    printf("\nThe value of i after for loop is %d", i);
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

i = 1
i = 2
i = 3
i = 4
i = 5
The value of i after for loop is 6
PS F:\code1\abc\output>
```

Case:3 two variable initialization in for loop

Case:4 two var, one without init

```c
#include<stdio.h>
int main(){
    int i, j;
    for(i = 1, j = 0; i<=5; i++){
        printf("\ni = %d , j = %d", i, j);
    }
    printf("\nThe value of i after for loop is %d and j is %d", i, j);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

i = 1 , j = 0
i = 2 , j = 0
i = 3 , j = 0
i = 4 , j = 0
i = 5 , j = 0
The value of i after for loop is 6 and j is 0
PS F:\code1\abc\output>
```

```c
#include<stdio.h>
int main(){
    int i, j;
    for(j = 0; i<=5; i++){
        printf("\ni = %d , j = %d", i, j);
    }
    printf("\nThe value of i after for loop is %d and j is %d", i, j);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

The value of i after for loop is 4169728 and j is 0
PS F:\code1\abc\output>
```
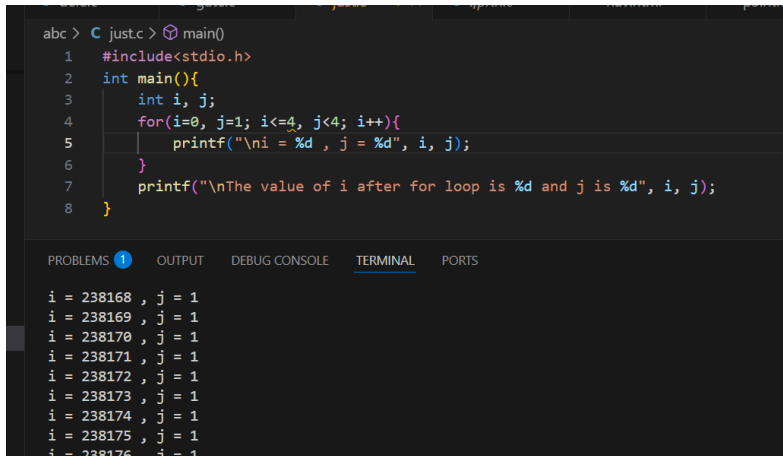
--expression-2—

Case:1 Infinite loop- no condition

--expression-2—

Case:2 Infinite loop- two condition in for loop, loop will execute until second becomes false(j<4)

```c
abc > C just.c > ⦿ main()
1    #include<stdio.h>
2    int main(){
3        int i, j;
4        for(i=0, j=1; i<=4, j<4; i++){
5            printf("\ni = %d , j = %d", i, j);
6        }
7        printf("\nThe value of i after for loop is %d and j is %d", i, j);
8    }
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
i = 238168 , j = 1
i = 238169 , j = 1
i = 238170 , j = 1
i = 238171 , j = 1
i = 238172 , j = 1
i = 238173 , j = 1
i = 238174 , j = 1
i = 238175 , j = 1
i = 238176 , j = 1
```

--expression-2—

Case:3 if logical operators are there then it is considered as a one condition and terminated when both individually become false

```c
#include<stdio.h>
int main(){
    int i, j;
    for(i=0, j=1; i<=100 || j<3 ; i++,j++){
        printf("\ni = %d , j = %d", i, j);
        //if (i == 4) break;
    }
    printf("\nThe value of i after for loop is %d and j is %d", i, j);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

i = 83 , j = 84
i = 84 , j = 85
i = 85 , j = 86
i = 86 , j = 87
i = 87 , j = 88
i = 88 , j = 89
i = 89 , j = 90
i = 90 , j = 91
i = 91 , j = 92
i = 92 , j = 93
i = 93 , j = 94
i = 94 , j = 95
i = 95 , j = 96
i = 96 , j = 97
i = 97 , j = 98
i = 98 , j = 99
i = 99 , j = 100
i = 100 , j = 101
The value of i after for loop is 101 and j is 102
PS F:\code1\abc\output>
```

--expression-2—

Case:4 in condition, assignment operator

```c
abc > C just.c > ⊗ main()
1    #include<stdio.h>
2    int main(){
3        int i, j;
4        for(i=0, j=1; i == 10 ; i++,j++){
5            printf("\ni = %d , j = %d", i, j);
6        }
7        printf("\nThe value of i after for loop is %d and j is %d", i, j);
8    }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

The value of i after for loop is 0 and j is 1
PS F:\code1\abc\output> 
```

--expression-2—

Case:5 in condition, 0 – treated as false, else will treated as true , false will not execute loop body, while true condition must be break!

```c
#include<stdio.h>
int main(){
    int i, j;
    for(i=0, j=1; 0 ; i++,j++){
        printf("\ni = %d , j = %d", i, j);
    }
    printf("\nThe value of i after for loop is %d and j is %d", i, j);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

The value of i after for loop is 0 and j is 1
PS F:\code1\abc\output>
```

```c
#include<stdio.h>
int main(){
    int i, j;
    for(i=0, j=1; 1; i++,j++){
        printf("\ni = %d , j = %d", i, j);
    }
    printf("\nThe value of i after for loop is %d and j is %d", i, j);
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

i = 35683 , j = 35684
i = 35684 , j = 35685
i = 35685 , j = 35686
i = 35686 , j = 35687
i = 35687 , j = 35688
i = 35688 , j = 35689
i = 35689 , j = 35690
```

--expression-2—

Case:2 in condition, 0 – treated as false, else will treated as true (25) , false will not execute loop body, while true condition must be break(I == 4)!

```c
abc > C just.c > ⊘ main()
1    #include<stdio.h>
2    int main(){
3        int i, j;
4        for(i=0, j=1; 25 ; i++,j++){
5            printf("\ni = %d , j = %d", i, j);
6            if (i == 4) break;
7        }
8        printf("\nThe value of i after for loop is %d and j is %d", i, j);
9    }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

i = 0 , j = 1
i = 1 , j = 2
i = 2 , j = 3
i = 3 , j = 4
i = 4 , j = 5
The value of i after for loop is 4 and j is 5
PS F:\code1\abc\output>
```

--expression-3 —

Case:1 More than one incremental/decremental

```c
abc > C just.c > ⊙ main()
1   #include<stdio.h>
2 ∨ int main(){
3       int i, j;
4 ∨     for(i=0, j=1; i<=100, j<4; i++,j++){
5           printf("\ni = %d , j = %d", i, j);
6       }
7       printf("\nThe value of i after for loop is %d and j is %d", i, j);
8   }
```

```
PROBLEMS 1    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

i = 0 , j = 1
i = 1 , j = 2
i = 2 , j = 3
The value of i after for loop is 3 and j is 4
PS F:\code1\abc\output>
```

;

Case: for();

For(); → empty loop &
Body will be executed once
after for statement.

```c
#include<stdio.h>
int main(){
    int i, j;
    for(i = 0, j = 1; i < 5, j < 6; j++);
    {
        printf("\ni = %d , j = %d", i, j);
        i++;
    }
    printf("\nThe value of i after for loop is %d and j is %d", i, j);
}
```

PROBLEMS 2   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'just.exe'

i = 0 , j = 6
The value of i after for loop is 1 and j is 6
PS F:\code1\abc\output>
```

;

Case: for();

For(); → empty loop &
Body will be executed once
after for statement.

```c
abc > C gotojump.c > ⊙ main()
1    #include<stdio.h>
2    int main(){
3        int i;
4        for(i = 0; i<10; i++);
5        printf("The value of i is: %d",i);
6    }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'gotojump.exe'
The value of i is: 10
PS F:\code1\abc\output>
```

--expression-3 —

Case: infinite loop


```c
abc > C just.c > ⊗ main()
1    #include<stdio.h>
2    int main(){
3        int i, j;
4        for(;;){
5            printf("\ni = %d , j = %d", i, j);
6            |
7        }
8        printf("\nThe value of i after for loop is %d and j is %d", i, j);
9    }
```

PROBLEMS ①   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
i = 3006464 , j = 4199120
i = 3006464 , j = 4199120
i = 3006464 , j = 4199120
i = 3006464 , j = 4199120
i = 3006464 , j = 4199120
i = 3006464 , j = 4199120
i = 3006464 , j = 4199120
```

```c
abc > C dem1.c > ...
1    #include <stdio.h>
2
3    int main() {
4        for (int i = 0; i < 3; i++) {
5            printf("Iteration %d: Address of i = %p\n", i, (void*)&i);
6        }
7        return 0;
8    }
9
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'dem1.exe'
Iteration 0: Address of i = 0061FF1C
Iteration 1: Address of i = 0061FF1C
Iteration 2: Address of i = 0061FF1C
PS F:\code1\abc\output>
```

Sum of 10 natural number:

```
10    int main(){
11      int sum = 0 , i;
12      for(i = 1; i<=10; i++){
13        sum = sum+i;
14      }
15      printf("sum of 5 natural n umber is: %d ", sum );
16
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'dem1.exe'
sum of 5 natural n umber is: 55
PS F:\code1\abc\output>
```

Sum of square of 5 natural number:

```
9
10   int main(){
11     int sum = 0 , i,n ;
12     printf("Enter the number:");
13     scanf("%d",&n);
14     for(i = 1; i<=n; i++){
15       sum = sum+i*i;
16     }
17     printf("sum of square of 5 natural n umber is: %d ", sum );
18
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'dem1.exe'
Enter the number:5
sum of square of 5 natural n umber is: 55
PS F:\code1\abc\output>
```

Cause immediate exit from the switch , while, do while and for loop

Common uses:
Escape early form the loop, in a switch statement .

```c
abc > C d.c > ...
1    #include <stdio.h>
2
3    int main() {
4        for (int i = 1; i <= 10; i++) {
5            if (i == 5) {
6                continue;  // Skip printing 5
7            }
8            if (i == 8) {
9                break;  // Stop the loop when i = 8
10           }
11           printf("%d ", i);
12       }
13   }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'d.exe'
1 2 3 4 6 7
PS F:\code1\abc\output>
```

Continue Statement:

Skips the remaining statements in a body of while, do while, for loop
Proceed with the next iteration of the loop

```c
abc > C dem1.c > ...
1   #include <stdio.h>|
2   int main() {
3     int i;
4
5     for (i = 0; i < 10; i++) {
6       if (i == 4) {
7         continue;
8       }
9       printf("%d\n", i);
10    }
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'dem1.exe'
0
1
2
3
5
6
7
8
9
PS F:\code1\abc\output> []
```

Continue Statement:

```c
abc > C dem1.c > ...
 1    #include <stdio.h>
 2
 3    int main() {
 4      int i = 0;
 5
 6      while (i < 10) {
 7        if (i == 4) {
 8          i++;  // Increment `i` before `continue`
 9          continue;  // Skips the rest of the loop and jumps to the next iteration
10        }
11        printf("%d\n", i);
12        i++;  // Normal increment for other cases
13      }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'dem1.exe'
0
1
2
3
5
6
7
8
9
PS F:\code1\abc\output>
```

Ina case of for loop

The loop structure **automatically increments i** after each iteration.

That's why no need for i++ before continue!

```c
abc > C pattern1.c > ⊙ main()
1    #include<stdio.h>
2    int main(){
3        int i,j;
4        for(i=0;i<3;i++){
5            for(j=0;j<5;j++){
6                printf("* ");
7            }
8            printf("\n");
9        }
10   }
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'pattern1.exe'
* * * * *
* * * * *
* * * * *
PS F:\code1\abc\output>
```

Pattern *:

```c
int main() {
    int i,j,n;
    scanf("%d",&n);

    for (i = 0; i<n; i++){
        for(j=0;j<=i;j++){
            printf("* ");
        }
        printf("\n");
    }
}
```

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'d.exe'
6
*
* *
* * *
* * * *
* * * * *
* * * * * *
PS F:\code1\abc\output>
```

```c
#include<stdio.h>
int main(){
    int i,j,n;
    scanf("%d",&n);

    for(i=0;i<n;i++){
        for(j=0;j<n-i;j++){
            printf("* ");
        }
        printf("\n");
    }
}
```

```
PS F:\code1> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'pattern1.exe'
5
* * * * *
* * * *
* * *
* *
*
PS F:\code1\abc\output>
```

```c
abc > C pattern1.c > ⊗ main()
  1    #include<stdio.h>
  2    int main(){
  3        int i,j,n;
  4        scanf("%d",&n);
  5
  6        for(i=0;i<n;i++){
  7            printf("%*s", i * 2, ""); //
  8            for(j=0;j<n-i;j++){
  9                printf("* ");
 10            }
 11            printf("\n");
 12        }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

      *
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'pattern1.exe'
5
* * * * *
  * * * *
    * * *
      * *
        *
```

printf("%*s", width, string);

*➔ take the width from a variable instead of a fixed number

Width (i*2)➔ The number of spaces allocated before the string

String("")➔Actual string to print

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        printf("%*s", i * 2, "");
        for (int j = 1; j <= n - i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }
}
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'pattern2.exe'
5
1 2 3 4 5
  1 2 3 4
    1 2 3
      1 2
        1
PS F:\code1\abc\output>
```

```c
#include <stdio.h>
int main() {
    int n, coef = 1;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        printf("%*s", (n - i) * 2, "");
        // Loop for each column (binomial coefficient)
        for (int j = 0; j <= i; j++) {
            // Compute binomial coefficient using iterative method
            if (j == 0 || i == 0)
                coef = 1;
            else
                coef = coef * (i - j + 1) / j;

            printf("%4d", coef);
        }
        printf("\n");
    }
}
```

PROBLEMS　　OUTPUT　　DEBUG CONSOLE　　**TERMINAL**　　PORTS

```
            1
         1   1
      1   2   1
   1   3   3   1
1   4   6   4   1
PS F:\code1\abc\output>
```

Let x be an integer which can take a value of 0 or 1. The statement

```
if (x == 0) x = 1; else x = 0;
```

is equivalent to which one of the following ?

A. $x = 1 + x$;
B. $x = 1 - x$;
C. $x = x - 1$;
D. $x = 1\%x$;

```
main()
{
    int x, y, m, n;
    scanf("%d %d", &x, &y);
    /* Assume x>0 and y>0*/
    m = x; n = y;
    while(m != n)
        {
            if (m > n)
                m = m-n;
            else
                n = n-m;
        }
    printf("%d", n);
}
```

The program computes

    A. $x + y$ using repeated subtraction

    B. $x \mod y$ using repeated subtraction

    C. the greatest common divisor of x and y

    D. the least common multiple of x and y

Consider the following program fragment for reversing the digits in a given integer to obtain a new integer.

Let $n = d_1 d_2 \ldots d_m$.

```c
int n, rev;
rev = 0;
while(n > 0) {
    rev = rev * 10 + n%10;
    n = n/10;
}
```

The loop invariant condition at the end of the ith iteration is:

A. $n = d_1 d_2 \ldots d_{m-i}$ **and** $rev = d_m d_{m-1} \ldots d_{m-i+1}$

B. $n = d_{m-i+1} \ldots d_{m-1} d_m$ **or** $rev = d_{m-i} \ldots d_2 d_1$

C. $n \neq rev$

D. $n = d_1 d_2 \ldots d_m$ **or** $rev = d_m \ldots d_2 d_1$

Pattern number

```c
abc > C patternum.c > ...
  1   #include <stdio.h>
  2
  3   int main() {
  4       int n = 5; // Number of rows
  5
  6       for (int i = 1; i <= n; i++) {
  7           // Print leading spaces using %*s
  8           printf("%*s", (n - i) + 1, "");
  9
 10           // Print numbers
 11           for (int j = 1; j <= (2 * i - 1); j++) {
 12               printf("%d", j);
 13           }
 14           printf("\n");
 15       }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> & .\'patternum.exe'
     1
    123
   12345
  1234567
 123456789
PS F:\code1\abc\output>
```

```c
abc > C patternalpha.c > ...
  1    #include <stdio.h>
  2
  3    int main() {
  4        int n = 5; // Number of rows
  5
  6        for (int i = 1; i <= n; i++) {
  7            // Print leading spaces using %*s
  8            printf("%*s", (n - i) + 1, "");
  9
 10            // Print alphabets
 11            for (int j = 0; j < (2 * i - 1); j++) {
 12                printf("%c", 'A' + j);
 13            }
 14            printf("\n");
 15        }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'patternalpha.exe'
     A
    ABC
   ABCDE
  ABCDEFG
 ABCDEFGHI
PS F:\code1\abc\output>
```

```c
abc > C pyramid.c > ⊕ main()
1   #include <stdio.h>
2
3   int main() {
4       int i, space, rows, k;
5
6       printf("Enter the number of rows: ");
7       scanf("%d", &rows);
8
9       for (i = 1; i <= rows; ++i) {
10          // Printing spaces before stars
11          for (space = 1; space <= (rows - i); ++space) {
12              printf("  ");
13          }
14          for (k = 0; k < 2 * i - 1; ++k) {
15              printf("* ");
16          }
17          printf("\n");
18      }
19  }
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> & .\'pyramid.exe'
Enter the number of rows: 6
          *
        * * *
      * * * * *
    * * * * * * *
  * * * * * * * * *
* * * * * * * * * * *
PS F:\code1\abc\output>
```

Pattern Inverted Pyramid

```c
abc > C inverted.c > main()
1    #include <stdio.h>
2    int main() {
3        int i, j, space, n = 5;
4        for (i = 0; i < n; i++) {
5            // Print leading spaces
6            for (space = 0; space < 2 * i; space++) {
7                printf(" ");
8            }
9            // Print stars
10           for (j = 0; j < (2* (n - i)) - 1; j++) {
11               printf("* ");
12           }
13           printf("\n");
14       }
15
16       return 0;
17   }
18
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS F:\code1\abc\output> cd 'f:\code1\abc\output'
PS F:\code1\abc\output> & .\'inverted.exe'
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
```

Patterns:
https://www.programiz.com/c-programming/examples/pyramid-pattern

https://www.studytonight.com/c-programs/c-program-to-print-hollow-square-pattern-program

https://www.simplilearn.com/tutorials/c-tutorial/c-pattern-programs