Introduction to Computing and Programming

Revision

Input & Output Question

What is Stdin: fgets()

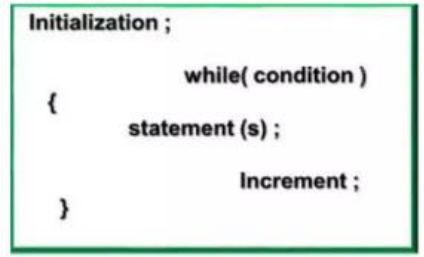
- **Stdin** (standard input) is defined in the <stdio.h> header file and represents the input stream that the C program **reads** from by default; Predefined file stream associated with input device typically **Keyboard;**
- Commonly used with input functions, **scanf(), fgets(), getc();**
  - **scanf(): Reads formatted input from stdin, Like an integer;**
  - **fgets(): Reads a string from stdin or line of text & store it in the buffer;**
  - **getc(): Reads a single character from stdin**

```
#include <stdio.h>
int main() {
    char buffer[50];
    printf("Enter a string: ");
    fgets(buffer, sizeof(buffer), stdin);  // Reads input from stdin
    printf("You entered: %s", buffer);
    return 0;
}
```

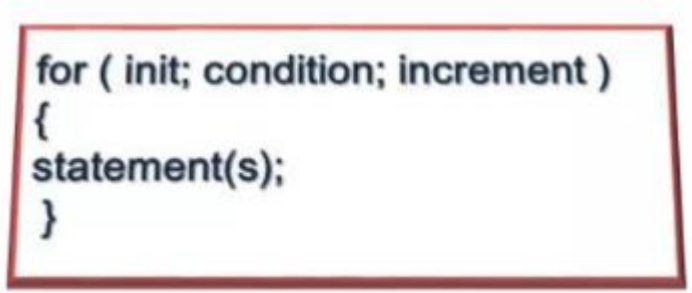
Loops Practice

Loops: When to use while & for loop?

- Use **while** when the loop should run until a specific condition is met, and the **number of iterations cannot be determined upfront**.
  - Eg: Number checking for **palindrome, Armstrong**;
  - Syntax:



- Use **for** **when the loop should run a set number of times** or when you have a counter-controlled loop with a clear start, stop, and step.
  - Eg: Pattern printing
  - Syntax:



Check whether a number is palindrome or not

```
#include <stdio.h>
int main() {
    int num, reversedNum = 0, remainder, originalNum;
    printf("Enter an integer: ");
    scanf("%d", &num);
    originalNum = num;
    while (num != 0) {
        remainder = num % 10;  // Get the last digit of the number
        reversedNum = reversedNum * 10 + remainder;  // Build the reversed number
        num = num / 10;  // Remove the last digit from the original number }
if (originalNum == reversedNum) {
        printf("%d is a palindrome.\n", originalNum);
    } else {
        printf("%d is not a palindrome.\n", originalNum);
    } return 0; }
```

```c
#include <stdio.h>
int main() {
    int num, originalNum, remainder, result = 0;
    printf("Enter a three-digit integer: ");
    scanf("%d", &num);
    originalNum = num;

    while (originalNum != 0) {
        remainder = originalNum % 10;
        result += remainder * remainder * remainder;
        originalNum /= 10;
    }
    if (result == num)
        printf("%d is an Armstrong number.", num);
    else
        printf("%d is not an Armstrong number.", num);
    return 0;
}
```

Write a C program to check whether a number is Armstrong number

- **Example:**
  - **153 is an Armstrong number because 1^3+5^3+3^3=153.**
  - **122 is not an Armstrong number** because 1^3+2^3+2^3= 1+8+8 = 17 which is not equal to 122.
  - 9474 = 9^4+4^4+7^4+4^4 is an Armstrong number.

```
#include <stdio.h>
#include<math.h>
int main() {
int n;
printf("Enter the number to check");
scanf("%d",&n);
int flag = 0;
 for (int i = 2; i<=sqrt(n); i++)    {
 if (n%i == 0)       {
printf("%d is not prime", n);
 flag = 1;
break;       }    }
if (flag == 0){    printf("%d is prime", n);}
return 0;}
```

Write a C program to check whether a number is prime number or not

Write a C program to find sum of n natural numbers

```
#include <stdio.h>
int main() {
    int n, sum = 0;
    printf("Enter a positive integer: "); // Input the value of n
    scanf("%d", &n); // Make sure the input is a positive integer
    if (n < 0) {
        printf("Invalid input! Please enter a positive integer.\n");
        return 0;}
    for (int i = 1; i <= n; i++) {
            sum += i;
        }
    printf("Sum of the first %d natural numbers is: %d\n", n, sum);
    return 0;}
```

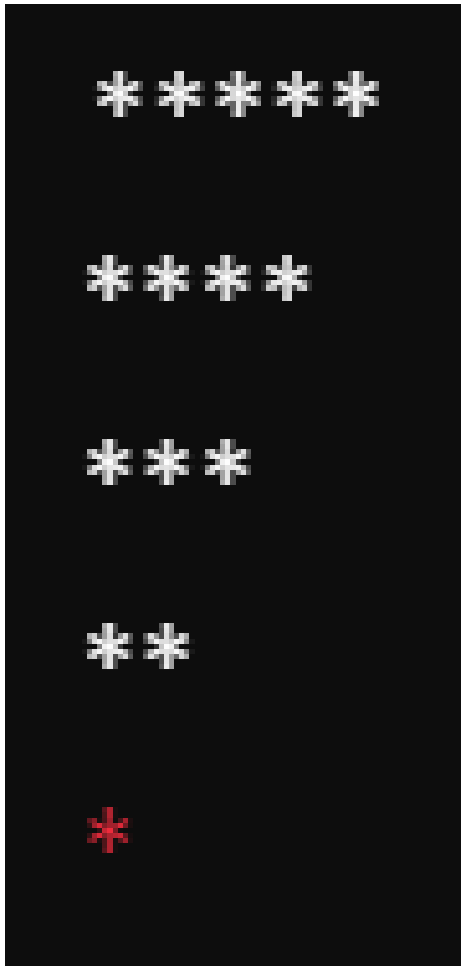
Write a C program to find the table of 2

```
#include <stdio.h>


int main() {
    int i;
    // Print the multiplication table of 2
    printf("Multiplication table of 2:\n");
    for (i = 1; i <= 10; i++) {
        printf("2 x %d = %d\n", i, 2 * i);
    }
    return 0;
}
```

```
#include <stdio.h>
int main() {
  int i, n;
  int t1 = 0, t2 = 1;
  int nextTerm = t1 + t2;
  printf("Enter the number of terms: ");
  scanf("%d", &n);
  printf("Fibonacci Series: %d, %d, ", t1, t2);
  for (i = 3; i <= n; ++i) {
    printf("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
    nextTerm = t1 + t2; }
  return 0;}
```

Fibonacci Sequence




```
#include <stdio.h>
int main() {
    int rows = 5;  // Number of rows in the
pattern

    // Outer loop for each row
    for (int i = rows; i >= 1; i--) {
        // Inner loop for printing stars
        for (int j = 1; j <= i; j++) {
            printf("*");
        }
        printf("\n");  // Move to the next line after
each row
    }
    return 0;
}
```

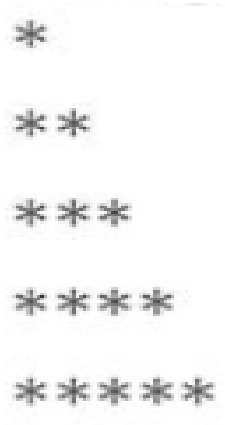
Write a C program to display a pyramid

```
#include <stdio.h>

int main() {
   int rows = 5;  // Number of rows for the pyramid

   // Outer loop to handle the number of rows
   for (int i = 1; i <= rows; i++) {
      // Inner loop to print stars for each row
      for (int j = 1; j <= i; j++) {
         printf("*");
      }
      // Move to the next line after printing each row
      printf("\n");
   }

   return 0;
}
```



Write a C program to display a pyramid

```
#include <stdio.h>


int main() {
   int rows = 5;  // Number of rows for the pyramid


   // Outer loop to handle the number of rows
   for (int i = 1; i <= rows; i++) {
      // Inner loop to print stars for each row
      for (int j = 1; j <= i; j++) {
         printf(“j");
      }
      // Move to the next line after printing each row
      printf("\n");
   }


   return 0;
}
```

```
1
12
123
1234
12345
```

```
#include<stdio.h>
int main()
{ int n;
    printf("Enter the value of n");
    scanf("%d",&n);
    for (int i =0; i < n; i++)
    { for(int j = 0; j<i; j++)
        { printf(" ");  }
        for (int k = 0; k < n-i; k++)
        {printf("*");}
       printf("\n");}
}
```

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

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

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

```
for(int i = 0; i<n-1; i++)
    {
        for(int j = 0; j < n-1-i; j++)
        {
            printf(" ");
        }
        for(int k = 0; k <=i; k++)
        {
            printf("*");
        }
       printf("\n");
    }
```

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

```
for(int i = 0; i<n-1; i++)
    {
        for(int j = 0; j < n-1-i; j++)
        {
            printf(" ");
        }
        for(int k = 0; k <=i; k++)
        {
            printf("*");
        }
        for(int k = 0; k <=i-1; k++)
        {
            printf("*");
        }
        printf("\n");
    }
```

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

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

for (int i =0; i < n; i++)    {
for(int j = 0; j<i; j++)        {           printf(" ");        }
for (int k = 0; k < n-i; k++)        {           printf("*");        }
for (int k = 0; k < n-i-1; k++)        {           printf("*");        }
printf("\n");    }
 }
```

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

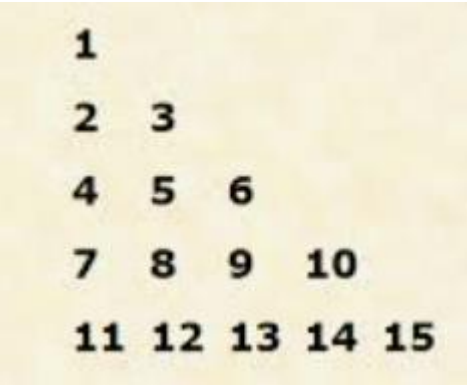
Diamond Pattern

Try printing These patterns

- Rectangle
- Hollo Rectangle
- Floyd’s Triangle
- Butterfly Pattern:

Hint: rows: 1 to n

space: 2*n-2*row



Precedence of an operator

Precedence	Operator	Description	Associativity
1	()	Parentheses (function call)	Left-to-Right
	[]	Array Subscript (Square Brackets)	
	.	Dot Operator	
	->	Structure Pointer Operator	
	++ , —	Postfix increment, decrement	
2	++ / —	Prefix increment, decrement	Right-to-Left
	+ / –	Unary plus, minus	
	! , ~	Logical NOT, Bitwise complement	
	(type)	Cast Operator	
	*	Dereference Operator	
	&	Addressof Operator	
	sizeof	Determine size in bytes	

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
8	&	Bitwise AND	Left-to-Right
9	^	Bitwise exclusive OR	Left-to-Right
10	|	Bitwise inclusive OR	Left-to-Right
11	&&	Logical AND	Left-to-Right
12	||	Logical OR	Left-to-Right
13	?:	Ternary conditional	Right-to-Left

Precedence	Operator	Description	Associativity
14	=	Assignment	Right-to-Left
	+= , -=	Addition, subtraction assignment	
	*= , /=	Multiplication, division assignment	
	%= , &=	Modulus, bitwise AND assignment	
	^= , |=	Bitwise exclusive, inclusive OR assignment	
	<<=, >>=	Bitwise shift left, right assignment	
15	,	comma (expression separator)	Left-to-Right

- int a = 5;

```
a++;      // Increment expression (a becomes 6)
a--;      // Decrement expression (a becomes 5)
```

Example of Expression

1. X=3+4*2=3+8=11

2. a= 3*4%5/2

=12%5/2

=2/2

=1

Example

```
#include <stdio.h>
int main(void) {
  int x = 10, y = 5;
  y = x++ + ++y;
  printf("x = %d y = %d", x, y);
  /* post incr ++ is has highest priority, so x becomes 11 but it'll increase only after the statement is evaluated, so it is not reflectred in the value of 'y' y = 10 + 5 x = x + 1 */
  return 0;
}
```

Let’s Solve

5/10*5+5*2

5|10&12>>2

10*4>>2 || 3

10/(5<10 && 20<30)

10/(5-5)