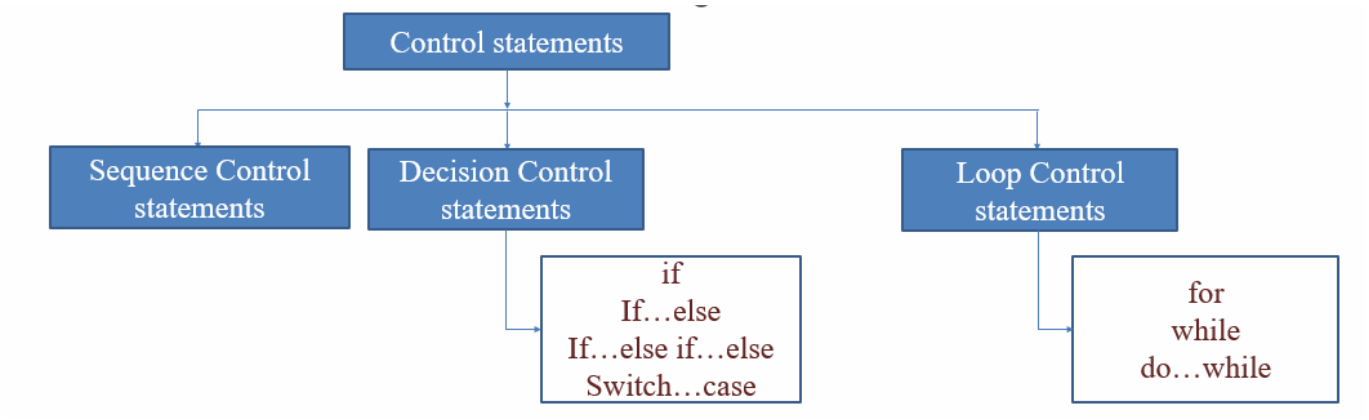
Unit 2



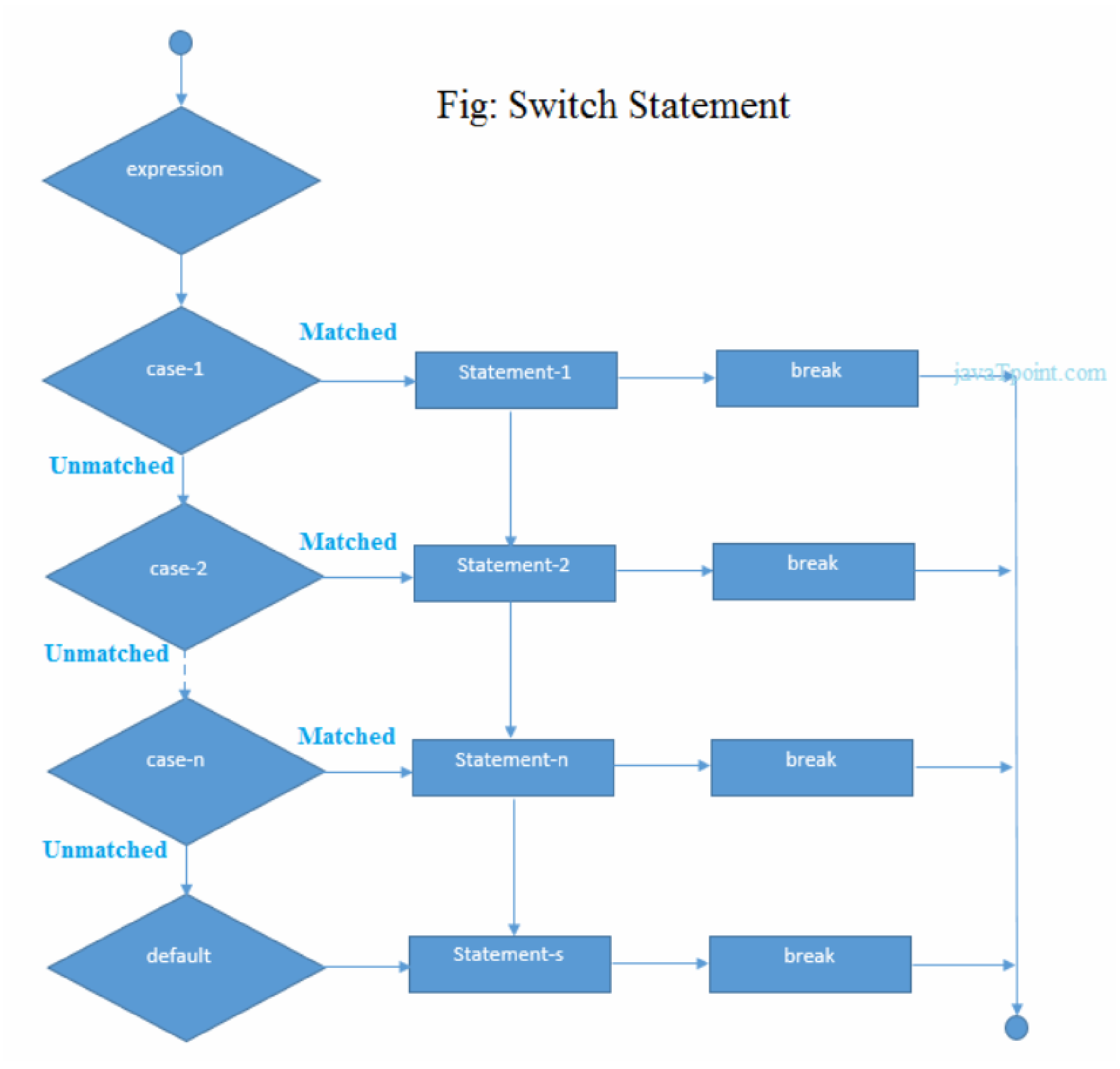
1. Introduction to Decision Control Statements
2. Conditional Branching Statements
    i. if Statement
    ii. if–Else Statement
    iii. Else–If Statement
    iv. switch-case
3. Iterative statements
    i. while
    ii. do-while
    iii. for
4. Loop control statements: break, continue, goto
5. Arrays
    i. Declaration of Arrays.
    ii. Accessing elements and Storing values in an array.
    iii. Operations on Arrays- Traversing, Inserting and Deletion of elements in an array.
    iv. Operations on Two Dimensional arrays.

Loops	

For loop
 	
Syntax :
 	
for(initialization: condition; updation)
{……………………..}
	Example:
	
	for (i=0 ; i<10 ; i++)
	{
	  printf (“%d\n”, i);
	}

While Loop (Entry controlled loop)
 	
Syntax :
 	
initialization;
while (condition)
{
  statements;
  updation;
}
	Example:
	
	i=0;
	while (i<10)
	{
	    printf (“%d\n”, i);
	    i++;
	}

Do While Loop (Exit controlled loop)
 	
Syntax :
 	
initialization;
do
{
    statements;
    updation;
} while (condition);
	Example:
	
	i=0;
	do
	{
	   printf (“%d\n”, i);
	    i++;
	} while (i<10);



Valid	Invalid
case 4-5	Duplicate cases Case 2-4: Case 4-6

1. Number Type Checker/ Positive, Negative, or Zero

```
#include <stdio.h>

 main()
{
  int myNum;
    printf("Enter a  number  ");
    scanf("%d", &myNum);
if (myNum > 0)
{
  printf("The value is a positive number.");
}
else if (myNum < 0)
{
  printf("The value is a negative number.");
}
else
{
  printf("The value is 0.");
}
}
```

```
Enter a  number  2
The value is a positive number.
```

2. Grade Classification Program:

```
#include <stdio.h>

int main() {
    int grade;


    printf("Enter your grade (0-100): ");
    scanf("%d", &grade);


    if (grade >= 90)
  {
        printf("Grade: A\n");
    }
  else if (grade >= 80)
  {
        printf("Grade: B\n");
    }
  else if (grade >= 70)
 {
        printf("Grade: C\n");
    }
  else if (grade >= 60)
   {
        printf("Grade: D\n");
    }
  else
  {
     printf("Grade: F\n");
   }
   return 0;
}
```

3. Time of Day Greeting Program:

```
#include <stdio.h>

int main()
{
   int hour;

   printf("Enter the current hour (0-23): ");
   scanf("%d", &hour);

   if (hour < 0 || hour > 23)
  {
     printf("Invalid hour entered.\n");
   }
  else if (hour < 12)
  {
     printf("Good morning!\n");
   }
  else if (hour < 18)
   {
     printf("Good afternoon!\n");
   }
  else
  {
     printf("Good evening!\n");
   }

   return 0;
}
```

4. Day of the Week

```
#include <stdio.h>

int main() {
   int day;

   printf("Enter a day number (1-7): ");
   scanf("%d", &day);

   switch (day)
{
     case 1:
        printf("Sunday\n");
        break;
     case 2:
        printf("Monday\n");
        break;
      case 3:
        printf("Tuesday\n");
        break;
      case 4:
        printf("Wednesday\n");
        break;
      case 5:
        printf("Thursday\n");
        break;
      case 6:
        printf("Friday\n");
        break;
      case 7:
        printf("Saturday\n");
        break;
      default:
        printf("Invalid day number.\n");
    }

    return 0;
}
```

5. Menu-Driven Calculator

```
#include <stdio.h>

int main() {
    char operator;
    double num1, num2, result;

    do {
      printf("Enter an operator (+, -, *, /) or 'q' to quit: ");
      scanf(" %c", &operator);

      if (operator == 'q')
      {
        break;
      }

      printf("Enter two numbers: ");
      scanf("%lf %lf", &num1, &num2);

      switch (operator)
      {
        case '+':
          result = num1 + num2;
          break;

        case '-':
          result = num1 - num2;
          break;

        case '*':
          result = num1 * num2;
          break;
```

```
           case '/':
              if (num2 != 0)
                 result = num1 / num2;
               else
                 printf("Error: Division by zero.\n");
                 break;
           default:
              printf("Invalid operator.\n");

        }

        printf("Result: %.2lf\n", result);

    } while (1);

    printf("Calculator terminated.\n");

    return 0;

}
```

Output:

```
Enter an operator (+, -, *, /) or 'q' to quit: *
Enter two numbers: 12
10
Result: 120.00
Enter an operator (+, -, *, /) or 'q' to quit: q
Calculator terminated.
```

6. Sum of N Natural Numbers:

```
#include <stdio.h>

int main() {
    int N, sum = 0;

    printf("Enter a value for N: ");
    scanf("%d", &N);

    for (int i = 1; i <= N; i++)
 {
        sum += i;
    }

    printf("Sum of the first %d natural numbers: %d\n", N, sum);

    return 0;
}
```

```
i=1;
while (i <= N)
  {
       sum =sum+i;
       i++;
    }
```

7. Multiplication Table

```
#include <stdio.h>

int main()
{
    int N;
    printf("Enter a number for the multiplication table: ");
    scanf("%d", &N);
    printf("Multiplication table for %d:\n", N);
    for (int i = 1; i <= 10; i++)
   {
        printf("%d  x  %d  =  %d\n", N, i, N * i);
    }
    return 0;
}
```

```
while (i <= 10)
{
        printf("%d x %d = %d\n", N, i, N * i);
        i++;
}
```

8. Factorial Calculation

```
#include <stdio.h>

int main() {
    int N, factorial = 1;

    printf("Enter a number for factorial calculation: ");
    scanf("%d", &N);

    for (int i = 1; i <= N; i++)
    {
        factorial = factorial * i;
    }

    printf("Factorial of %d: %d\n", N, factorial);

    return 0;
}
```

9. Even or Odd Sum Calculator (if-else and Loop):

```
#include <stdio.h>

int main() {
    int n, i;
```

```
   int sume = 0,sumo=0;

   printf("Enter the value of n: ");
   scanf("%d", &n);

   for (i = 1; i <= n; i++)
  {
      if (i % 2 == 0)
      {
         sume = sume+i;
      }
      else
      {
         sumo= sumo+ i;
      }
   }

   printf("Sum of even numbers from 1 to %d: %d\n", n, sume);
   printf("Sum of odd numbers from 1 to %d: %d\n", n, sumo);
   return 0;
}
```

Output:

```
Enter the value of n: 5
Sum of even numbers from 1 to 5: 6
Sum of odd numbers from 1 to 5: 9
```

10. Infinite Loop

```
#include <stdio.h>
 int main ()
{
  for( ; ; )
{
     printf("This loop will run forever.\n");
  }

  return 0;
}
```

11. Power Calculation

```
#include <stdio.h>

int main()
{
   int base, exponent;
   long result = 1;

   printf("Enter base and exponent: ");
   scanf("%d%d", &base, &exponent);
   while (exponent != 0)
   {
      result = result *base;
      --exponent;
   }

   printf("Result: %ld\n", result);

   return 0;
}
```

12.Fibonacci Series

```
#include <stdio.h>

int main() {
   int n, first = 0, second = 1, next;

   printf("Enter the number of terms: ");
   scanf("%d", &n);

   printf("Fibonacci Series:\n");

   while (n > 0)
{
      printf("%d, ", first);
      next = first + second;
      first = second;
      second = next;
      n--;
   }

   printf("\n");

   return 0;
}
```

Output:

```
Enter the number of terms: 5
Fibonacci Series:
0, 1, 1, 2, 3,
```

13.Numbers in Reverse order

```
#include <stdio.h>

int main()
{
   int N;

   printf("Enter a value for N: ");
   scanf("%d", &N);

   do
{
      printf("%d ", N);
      N--;
   } while (N >= 1);

   return 0;
}
```

Output:

```
Enter a value for N: 9
9 8 7 6 5 4 3 2 1
```

14.Reverse of a multi digit number

```
#include <stdio.h>

int main()
{
    int number, n, rev = 0;

    printf("Enter a two-digit number: ");
    scanf("%d", &number);

    // Store the original number
    n = number;

    // Reverse the digits
    while (number != 0)
    {
        int digit = number % 10;
        rev = rev * 10 + digit;
        number /= 10;
    }

    // Output the reversed number
    printf("The reverse of %d is: %d\n", n, rev);

    return 0;
}
```

Output:

```
Enter a two-digit number: 45678
The reverse of 45678 is: 87654
```

```
int main()
{
    int i;

    // Loop with only initialization and condition
    for (i = 1; i <= 5;)
    {
        printf("%d ", i);
       // i++;                          // Increment inside the loop body
    }

    return 0;
}
```

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

```
int main() {
    int i = 1;    // int i;

    // Loop with only condition and update
    for (; i <= 5; i++)
      {
        printf("%d ", i);
    }

    return 0;
}
```

```
#include <stdio.h>

int main()
 {
    int i = 1;

    // Loop with only update
    for (; ; i++)
   {
        if (i > 5)
        {
            break;  // Exit the loop when i exceeds 5
        }
        printf("%d ", i);
    }

    return 0;
}
```

Nested for loops

Multiplication Table

```
#include <stdio.h>

int main()
 {
    int rows = 5;
    int cols = 10;

    // Nested loop to print a multiplication table
    for (int i = 1; i <= rows; i++)
    {
        for (int j = 1; j <= cols; j++)
        {
            printf("%3d ", i * j);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
1  2  3  4  5  6  7  8  9 10
  2  4  6  8 10 12 14 16 18 20
  3  6  9 12 15 18 21 24 27 30
  4  8 12 16 20 24 28 32 36 40
  5 10 15 20 25 30 35 40 45 50
```

Printing a Square

```c
#include <stdio.h>

int main()
{
    int side = 4;

    // Nested loop to print a square
    for (int i = 1; i <= side; i++) {
        for (int j = 1; j <= side; j++) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

Output:

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

Displaying a Half-Pyramid Pattern

```c
#include <stdio.h>

int main() {
    int rows = 5;

    // Nested loop to print a half-pyramid pattern
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
1
1 2
1 2 3
```

```
1 2 3 4
1 2 3 4 5
```

Hollow Rectangle Pattern

```
#include <stdio.h>

int main() {
   int rows = 4;
   int cols = 6;

   // Nested loop to print a hollow rectangle pattern
   for (int i = 1; i <= rows; i++) {
      for (int j = 1; j <= cols; j++) {
         if (i == 1 || i == rows || j == 1 || j == cols) {
            printf("* ");
         } else {
            printf("  ");
         }
      }
      printf("\n");
   }

   return 0;
}
```

Output:

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

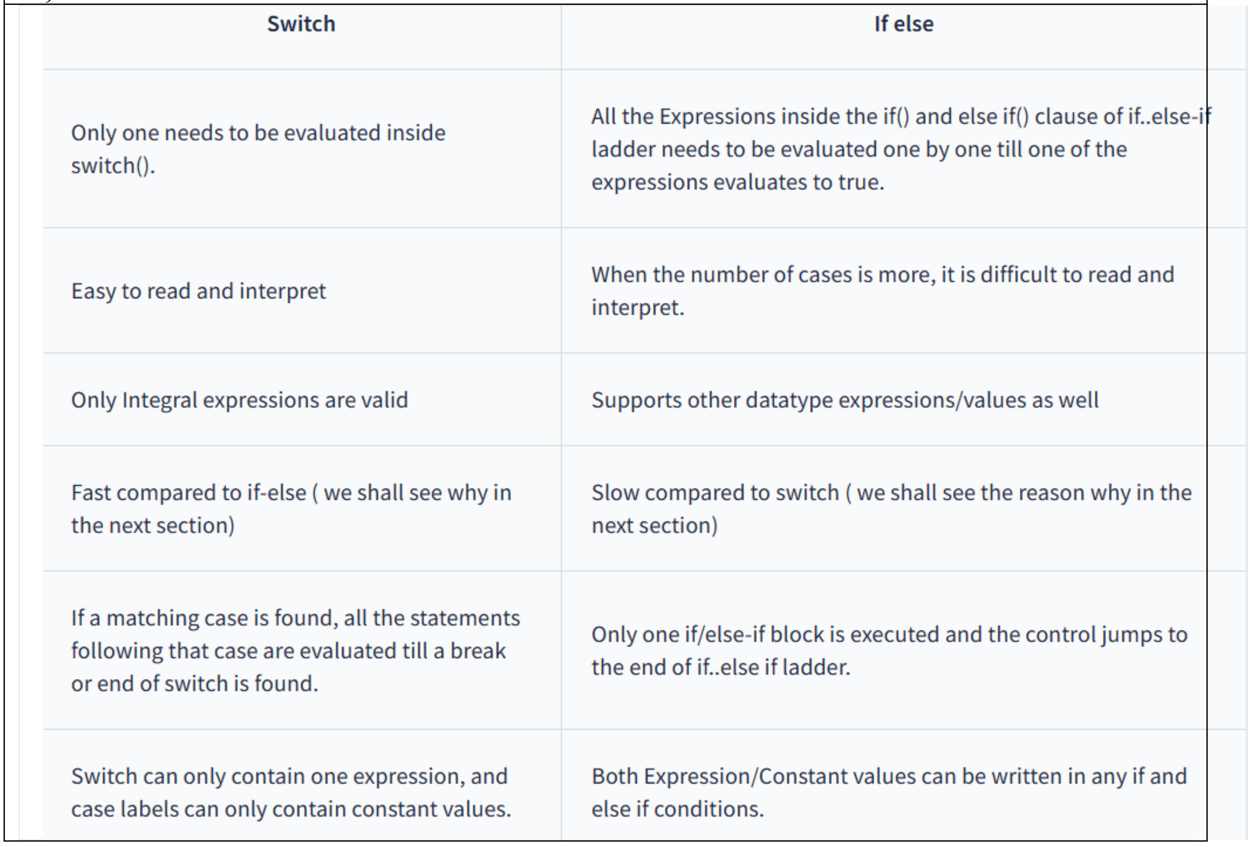
Print whether a given alphabet is vowel or constant

```
#include <stdio.h>

int main() {
   char alphabet;

   printf("Enter an alphabet: ");
   scanf(" %c", &alphabet);

   alphabet = tolower(alphabet);

   // Check if the entered character is an alphabet
   if ((alphabet >= 'a' && alphabet <= 'z') || (alphabet >= 'A' && alphabet <= 'Z'))
 {

      switch (alphabet)
{
         case 'a':
         case 'e':
         case 'i':
         case 'o':
         case 'u':
                printf("%c is a vowel.\n", alphabet);
                break;
            default:
                printf("%c is a consonant.\n", alphabet);
        }
    }
else
{
         printf("Invalid input. Please enter an alphabet.\n");
    }

    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int x = 10, y = 5;
    switch(x>y && x+y>0)
    {
        case 1:
        printf("hi");

        case 0:
        printf("bye");

        default:
        printf(" Hello bye ");
    }
}
```

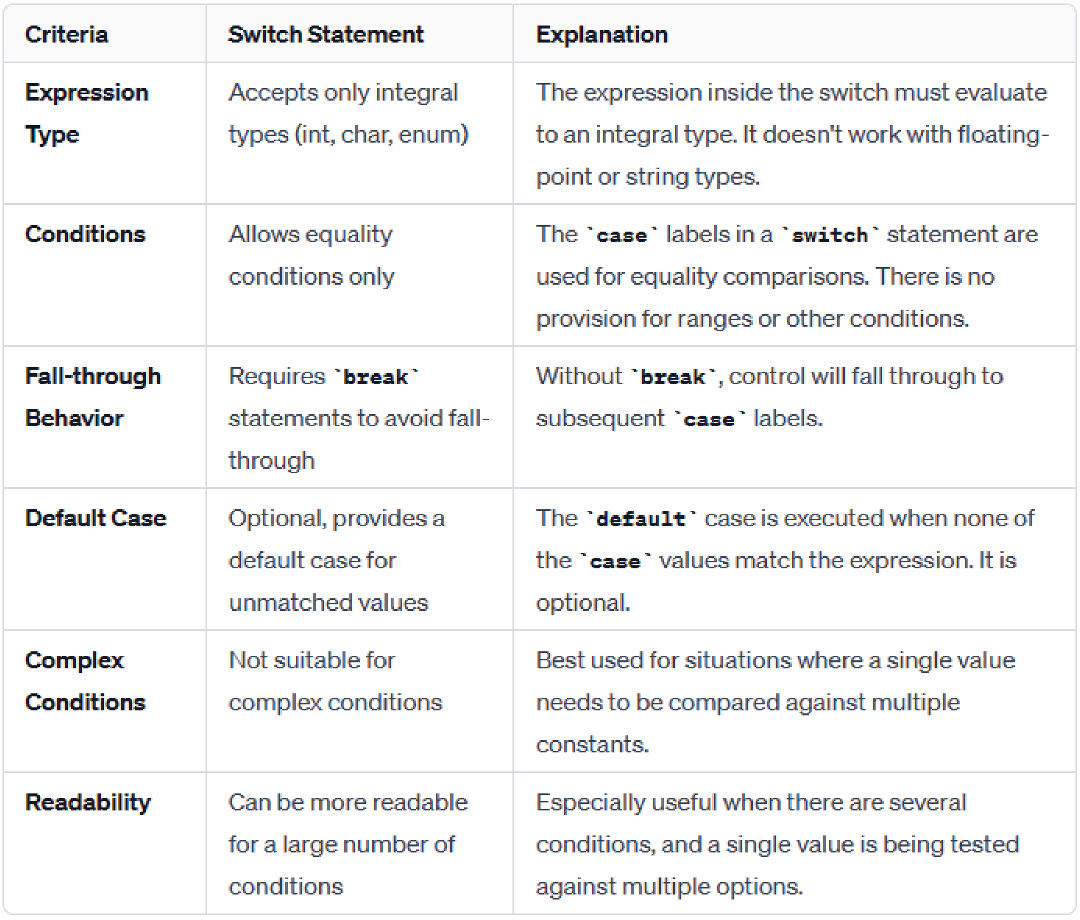
Output:
hibye Hello bye

```
#include <stdio.h>
int main()
{
   int a = 5;
int b = 10;
switch (a)
{ //Outer switch

    case 25 / 5:
        switch (b)
        { // Inner switch.
            case 100 / 10:
                printf("I am inside two switches!!\n");
                break;
            default:
                printf("me too!\n");
        }
        break;
    default:
        printf("I am default\n");
}

}
```

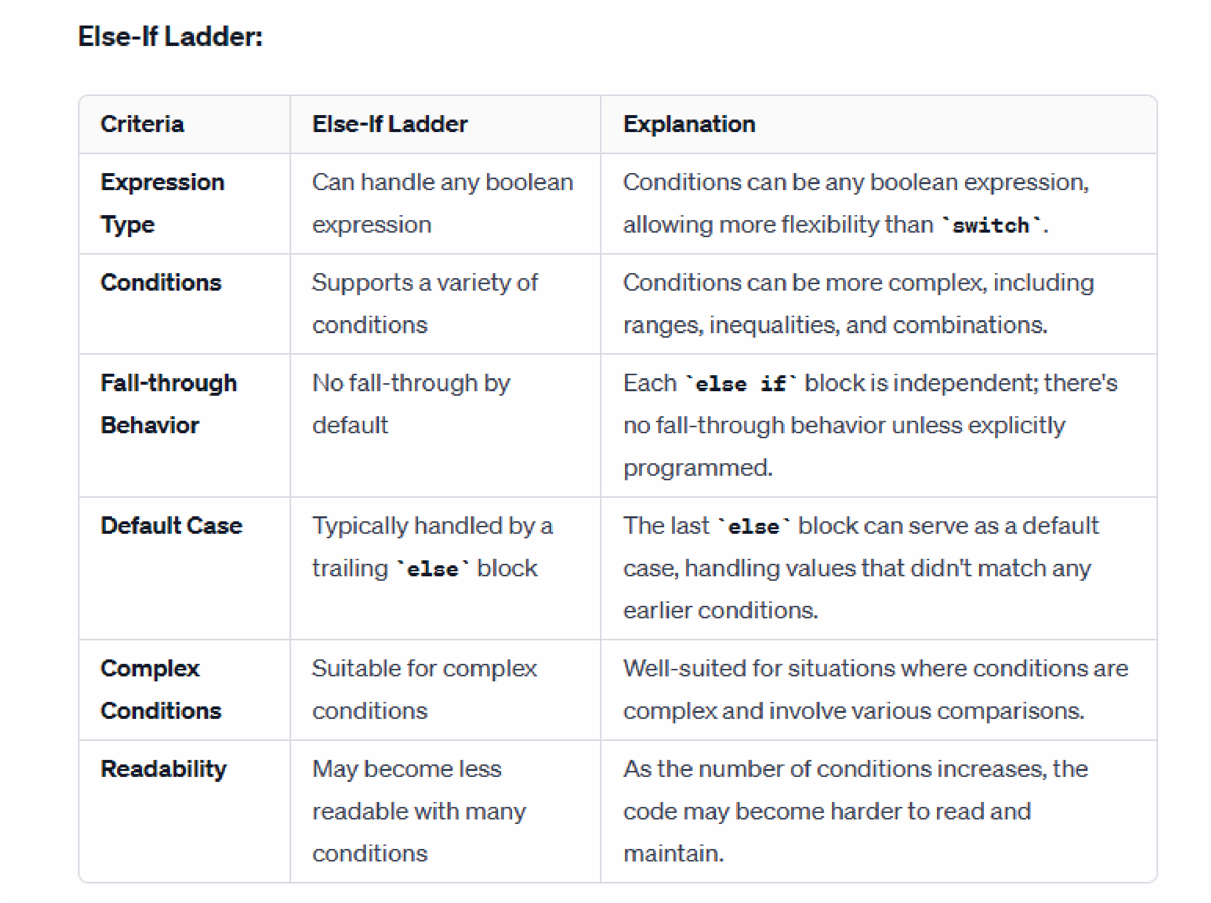


Switch	If else
Only one needs to be evaluated inside switch().	All the Expressions inside the if() and else if() clause of if..else-if ladder needs to be evaluated one by one till one of the expressions evaluates to true.
Easy to read and interpret	When the number of cases is more, it is difficult to read and interpret.
Only Integral expressions are valid	Supports other datatype expressions/values as well
Fast compared to if-else (we shall see why in the next section)	Slow compared to switch (we shall see the reason why in the next section)
If a matching case is found, all the statements following that case are evaluated till a break or end of switch is found.	Only one if/else-if block is executed and the control jumps to the end of if..else if ladder.
Switch can only contain one expression, and case labels can only contain constant values.	Both Expression/Constant values can be written in any if and else if conditions.

Switch Statement:



Criteria	Switch Statement	Explanation
Expression Type	Accepts only integral types (int, char, enum)	The expression inside the switch must evaluate to an integral type. It doesn't work with floating-point or string types.
Conditions	Allows equality conditions only	The `case` labels in a `switch` statement are used for equality comparisons. There is no provision for ranges or other conditions.
Fall-through Behavior	Requires `break` statements to avoid fall-through	Without `break`, control will fall through to subsequent `case` labels.
Default Case	Optional, provides a default case for unmatched values	The `default` case is executed when none of the `case` values match the expression. It is optional.
Complex Conditions	Not suitable for complex conditions	Best used for situations where a single value needs to be compared against multiple constants.
Readability	Can be more readable for a large number of conditions	Especially useful when there are several conditions, and a single value is being tested against multiple options.



Else-If Ladder:

Criteria	Else-If Ladder	Explanation
Expression Type	Can handle any boolean expression	Conditions can be any boolean expression, allowing more flexibility than `switch`.
Conditions	Supports a variety of conditions	Conditions can be more complex, including ranges, inequalities, and combinations.
Fall-through Behavior	No fall-through by default	Each `else if` block is independent; there's no fall-through behavior unless explicitly programmed.
Default Case	Typically handled by a trailing `else` block	The last `else` block can serve as a default case, handling values that didn't match any earlier conditions.
Complex Conditions	Suitable for complex conditions	Well-suited for situations where conditions are complex and involve various comparisons.
Readability	May become less readable with many conditions	As the number of conditions increases, the code may become harder to read and maintain.

```
#include <stdio.h>

main()
{
   for (int i = 0; i < 5; i++)
  {
      printf("%d ", i);

      if (i == 2)
         break;
   }
}
```

```
#include <stdio.h>

main()
{
   for (int i = 0; i < 5; i++)
  {
      if (i == 2)
      {
         continue;
      }
      printf("%d ", i);
   }
}
```

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

```
main()
{
   for (int i = 1; i < 50; i++)
  {
      if (i % 2!=0)
      {
         continue;
      }

      printf("%d ", i);
   }

}
```

Backward goto statement

```
#include <stdio.h>
main()
 {
   int a;
   startLoop:
   printf("\nEnter a\n ");
   scanf("%d",&a);
   printf("%d ", a);
      if (a > 2)
     {
         printf("\n%d is > 2\n ", a);
         goto startLoop;
      }
   printf("\nThis won't be printed.\n");
}
```

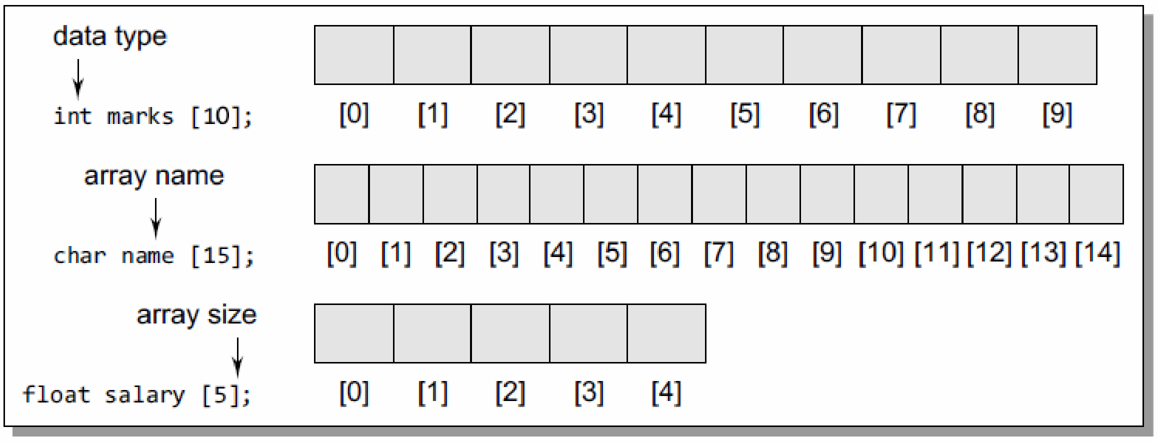
Output:

Forward goto statement

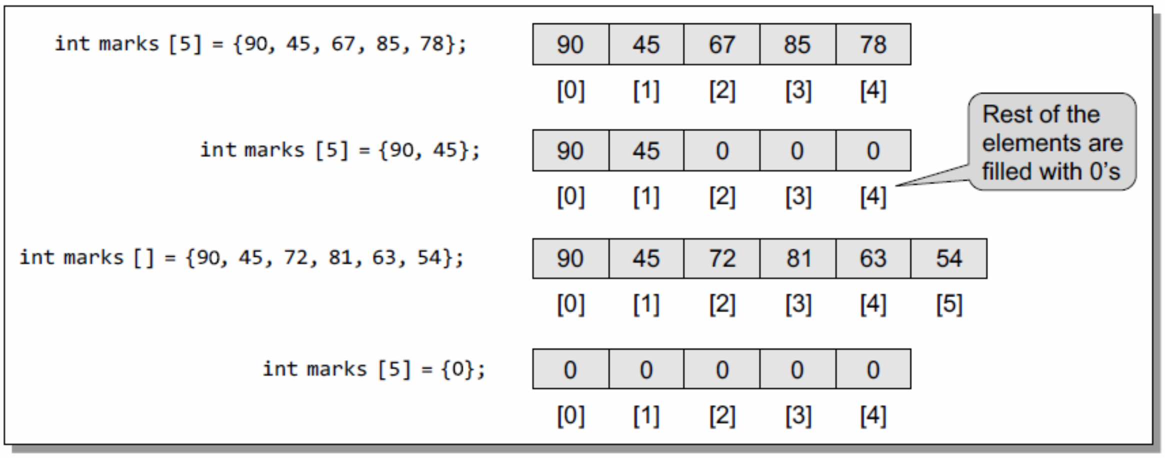
```
#include <stdio.h>
main()
{
   int i;
   for (i = 0; i < 5; i++)
  {
      printf("%d ", i);
           if (i == 2)
                goto endLoop;
   }
   printf("\nThis won't be printed in the
loop.\n");
    endLoop:
   printf("\nLoop ended.\n");
}
```

Output:

Arrays



Declaring arrays of different data types and sizes


int marks [5] = {90, 45, 67, 85, 78};
90 45 67 85 78
int marks [5] = {90, 45};
90 45 0 0 0
Rest of the elements are filled with 0's
int marks [] = {90, 45, 72, 81, 63, 54};
90 45 72 81 63 54
int marks [5] = {0};
0 0 0 0 0

Initialization of array elements

```
#include <stdio.h>
int main()
{
int i, a[5]={1,2,5,6,7};
for(i=0;i<5;i++)
printf("%d ", a[i]);
return 0;
}
```

```
#include <stdio.h>

int main()
{
int i, n, arr[20];
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
printf("\n Enter the elements of the array : ");
for(i=0; i<n; i++)
{
//printf("\n arr[%d] = ", i);
scanf("%d",&arr[i]);
}
printf("\n The array elements are ");
  for(i=0; i<n; i++)
      printf("\t %d", arr[i]);
return 0;
}
```

Output:

```
Enter the number of elements in the array : 5
arr[0] = 3
arr[1] = 2
arr[2] = 6
arr[3] = 7
arr[4] = 32
The array elements are          3       2       6       7       32
```

Linear Search

```
#include <stdio.h>
int main()
{
   int a[10], i,flag, key,n;
   printf("\nEnter number of elements of an array:\n");
   scanf("%d",&n);

   printf("\nEnter elements: \n");
   for (i=0; i<n; i++)
      scanf("%d", &a[i]);

   printf("\nEnter key to search: ");
   scanf("%d", &key);

   for (i=0; i<n; i++)
   {
      if (key == a[i])
      {
         printf("\nkey found at position %d", i+1);
         break;
      }
}

printf("\nkey does not exist.");

   return 0;
}
```

Binary Search

```
#include <stdio.h>
int main()
{
int i, low, high, mid, n, key, array[100];
printf("Enter number of elements");
scanf("%d",&n);
printf("Enter %d integers\n", n);
for(i = 0; i < n; i++)
scanf("%d",&array[i]);
printf("Enter value to find\n");
scanf("%d", &key);
low = 0;
high = n - 1;
mid = (low+high)/2;
while (low <= high)
{
    if(array[mid] < key)
            low = mid + 1;
     else if (array[mid] == key)
    {
        printf("%d found at location %d\n", key, mid+1);
        break;
     }
    else
          high = mid - 1;
  mid = (low + high)/2;
}
if(low > high)
printf("%d isn't present in the list\n", key);
return 0;
}
```

Calculate the sum and average of elements

```
#include <stdio.h>

int main() {
   int numbers[5];
   printf("Enter 5 integers:\n");

   // Input values into the array
   for (int i = 0; i < 5; i++)
   {
      printf("Element %d: ", i + 1);
      scanf("%d", &numbers[i]);
   }

   // Calculate the sum of elements
   int sum = 0;
   for (int i = 0; i < 5; i++)
   {
      sum =sum+ numbers[i];
   }

   // Calculate the average
   float average = (float)sum / 5;


   printf("\nEntered Elements:\n");
   for (int i = 0; i < 5; i++)
   {
      printf("Element %d: %d\n", i + 1, numbers[i]);
   }
```

```
    // Display the sum and average
    printf("\nSum: %d\n", sum);
    printf("Average: %.2f\n", average);

    return 0;
}
```

Output:

```
Enter 5 integers:
Element 1: 3
Element 2: 2
Element 3: 6
Element 4: 2
Element 5: 3
Entered Elements:
Element 1: 3
Element 2: 2
Element 3: 6
Element 4: 2
Element 5: 3

Sum: 16
Average: 3.20
```

Finding Maximum Element

```
#include <stdio.h>

int main() {
    // Declare an array of integers
    int numbers[5];

    printf("Enter 5 integers:\n");

    // Input values into the array
    for (int i = 0; i < 5; i++) {
        scanf("%d", &numbers[i]);
    }

    // Find the maximum element
    int max = numbers[0];
    for (int i = 1; i < 5; i++)
{
        if (numbers[i] > max) {
            max = numbers[i];
        }
    }

    // Display the entered elements with titles
    printf("\nEntered Elements:\n");
    for (int i = 0; i < 5; i++) {
        printf("Element %d: %d\n", i + 1, numbers[i]);
    }
    // Display the maximum element with title
    printf("\nMaximum Element: %d\n", max);

    return 0;
}
```

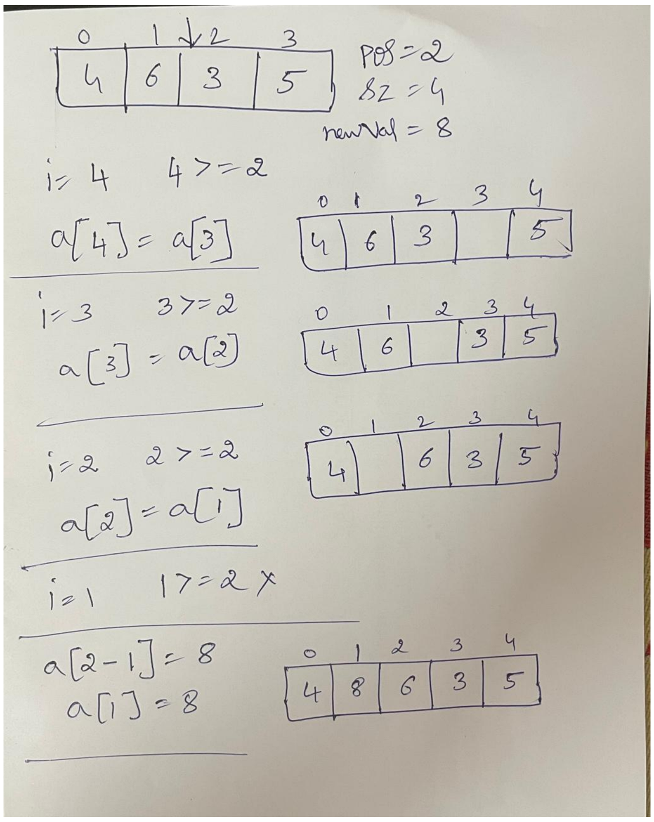
Operations on arrays

Traversing an Array:

```
#include <stdio.h>

int main() {
    int size;

    // Input the size of the array
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int arr[size];

    // Input array elements
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Traversing the array and printing elements
    printf("Array elements: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Inserting an Element in an Array:

```
#include <stdio.h>

int main() {
    int size, position, newValue;

    // Input the size of the array
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    int arr[size + 1];  // Creating a larger array to accommodate the new element

    // Input array elements
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }
    // Input position and value to insert
    printf("Enter the position to insert: ");
    scanf("%d", &position);
    printf("Enter the new value: ");
    scanf("%d", &newValue);

    // Shifting elements to make space for the new element
    for (int i = size; i >= position; i--)
    {
        arr[i] = arr[i - 1];
    }
```



```
    // Inserting the new value
    arr[position - 1] = newValue;

    // Printing the updated array
    printf("Array after insertion: ");
    for (int i = 0; i <= size; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

```
Output:
Enter the size of the array: 5
Enter the elements of the array:
3
2
```

```
4
5
6
Enter the position to insert: 2
Enter the new value: 100
Array after insertion: 3 100 2 4 5 6
```

Deleting an Element from an Array:

```
#include <stdio.h>

int main() {
   int size, position;

   // Input the size of the array
   printf("Enter the size of the array: ");
   scanf("%d", &size);

   int arr[size];

   // Input array elements
   printf("Enter the elements of the array:\n");
   for (int i = 0; i < size; i++) {
      scanf("%d", &arr[i]);
   }

   // Input position to delete
   printf("Enter the position to delete: ");
   scanf("%d", &position);

   // Shifting elements to remove the specified element
   for (int i = position - 1; i < size - 1; i++) {
      arr[i] = arr[i + 1];
   }

   // Decreasing the size of the array
   size--;

   // Printing the updated array
   printf("Array after deletion: ");
   for (int i = 0; i < size; i++) {
      printf("%d ", arr[i]);
   }

   return 0;
}
```

Output:

```
Enter the size of the array: 5
Enter the elements of the array:
3
```

```
5
7
8
9
Enter the position to delete: 1
Array after deletion: 5 7 8 9
```

2Dimensional arrays

```
#include <stdio.h>

int main()
{
int arr[2][2] = {12, 34, 56,32};
int i, j;
for(i=0;i<2;i++)
{
     printf("\n");
    for(j=0;j<2;j++)
        printf("%d\t", arr[i][j]);
}
return 0;
}
```

Output:

```
12      34

56      32
```

Read and print 2D array

```
#include <stdio.h>

int main()
{
   // Declare a 2D array
   int rows, cols;


   printf("Enter the number of rows: ");
   scanf("%d", &rows);

   printf("Enter the number of columns: ");
   scanf("%d", &cols);

   int matrix[rows][cols];

   // Prompt the user to enter elements for the array
   printf("Enter elements for the 2D array:\n");

   // Input values into the 2D array
   for (int i = 0; i < rows; i++)
   {
      for (int j = 0; j < cols; j++)
      {
         printf("Element [%d][%d]: ", i, j);
```

```
            scanf("%d", &matrix[i][j]);
        }
    }

    // Display the 2D array
    printf("\nEntered 2D Array:\n");
    for (int i = 0; i < rows; i++)
    {
         for (int j = 0; j < cols; j++)
        {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
Enter the number of rows: 2
Enter the number of columns: 2
Enter elements for the 2D array:
Element [0][0]: 1
2Element [0][1]:
2
Element [1][0]: 3
Element [1][1]: 4
Entered 2D Array:
1       2

3       4
```

Adding Two Matrices

```
#include <stdio.h>

int main() {
    int rows, cols;

    // Input the number of rows and columns
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int matrix1[rows][cols], matrix2[rows][cols], result[rows][cols];

    // Input elements into the first matrix
    printf("Enter the elements of the first matrix:\n");
    for (int i = 0; i < rows; i++)
   {
        for (int j = 0; j < cols; j++)
    {
            scanf("%d", &matrix1[i][j]);
        }
    }

    // Input elements into the second matrix
    printf("Enter the elements of the second matrix:\n");
    for (int i = 0; i < rows; i++)
 {
        for (int j = 0; j < cols; j++)
      {
            scanf("%d", &matrix2[i][j]);
        }
    }

    // Adding the two matrices
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
       {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }

    // Displaying the result matrix
    printf("Resultant matrix after addition:\n");
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
       {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
Enter the number of rows: 2
Enter the number of columns: 2
Enter the elements of the first matrix:
2
2
2
2
Enter the elements of the second matrix:
2
2
2
2
Resultant matrix after addition:
4       4
4       4
```

Multiplying Two Matrices

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

```
int main() {
   int rows1, cols1, rows2, cols2;

   // Input dimensions of the first matrix
   printf("Enter the number of rows for the first matrix: ");
   scanf("%d", &rows1);
   printf("Enter the number of columns for the first matrix: ");
   scanf("%d", &cols1);

   // Input dimensions of the second matrix
   printf("Enter the number of rows for the second matrix: ");
   scanf("%d", &rows2);
   printf("Enter the number of columns for the second matrix: ");
   scanf("%d", &cols2);

   if (cols1 != rows2) {
      printf("Error: Matrices cannot be multiplied.\n");
      return 1;
   }

   int matrix1[rows1][cols1], matrix2[rows2][cols2], result[rows1][cols2];

   // Input elements into the first matrix
   printf("Enter the elements of the first matrix:\n");
   for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols1; j++) {
         scanf("%d", &matrix1[i][j]);
      }
   }

   // Input elements into the second matrix
   printf("Enter the elements of the second matrix:\n");
   for (int i = 0; i < rows2; i++) {
      for (int j = 0; j < cols2; j++) {
         scanf("%d", &matrix2[i][j]);
      }
   }

   // Multiplying the two matrices
   for (int i = 0; i < rows1; i++)
   {
      for (int j = 0; j < cols2; j++)
     {
         result[i][j] = 0;
         for (int k = 0; k < cols1; k++)
         {
            result[i][j] += matrix1[i][k] * matrix2[k][j];
         }
      }
   }

   // Displaying the result matrix
   printf("Resultant matrix after multiplication:\n");
   for (int i = 0; i < rows1; i++) {
      for (int j = 0; j < cols2; j++) {
            printf("%d\t", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
Enter the number of rows for the first matrix: 2
Enter the number of columns for the first matrix: 2
Enter the number of rows for the second matrix: 2
Enter the number of columns for the second matrix: 2
Enter the elements of the first matrix:
1
2
1
2
Enter the elements of the second matrix:
1
2
1
2
Resultant matrix after multiplication:
3       6
3       6
```

Transposing a Matrix:

```
#include <stdio.h>

int main() {
    int rows, cols;

    // Input the number of rows and columns
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int matrix[rows][cols], transpose[cols][rows];

    // Input elements into the matrix
    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    // Transposing the matrix
    for (int i = 0; i < cols; i++) {
        for (int j = 0; j < rows; j++) {
            transpose[i][j] = matrix[j][i];
        }
    }

    // Displaying the transposed matrix
    printf("Transposed matrix:\n");
    for (int i = 0; i < cols; i++) {
        for (int j = 0; j < rows; j++) {
            printf("%d\t", transpose[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
Enter the number of rows: 2
Enter the number of columns: 3
Enter the elements of the matrix:
1
1
2
3
4
5
Transposed matrix:
1       3
1       4
2       5
```