DATA TYPES:

1) Declaring and Printing Variables

Algorithm

1. Declare an integer, a float, and a character variable.

2. Assign values to them.

3. Print their values.

Flowchart

Start → Declare int, float, char → Assign values → Print values → End

C Code

```
#include <stdio.h>

int main() {
    int num = 10;
    float fnum = 5.5;
    char ch = 'A';

    printf("Integer: %d\n", num);
    printf("Float: %.2f\n", fnum);
    printf("Character: %c\n", ch);

    return 0;
}
```

---

2) Convert Integer to Float

Algorithm:

1. Take an integer input from the user.

2. Convert it to a float.

3. Print both values.

Flowchart

Start → Input integer → Convert to float → Print both → End

C Code

```c
#include <stdio.h>

int main() {
    int num;
    float fnum;

    printf("Enter an integer: ");
    scanf("%d", &num);

    fnum = (float)num;

    printf("Integer: %d, Float: %.2f\n", num, fnum);

    return 0;
}
```

---

3)Print Size of Data Types

Algorithm

1. Use sizeof operator on different data types.

2. Print their sizes.

Flowchart

Start → Use sizeof() on int, float, char, double, short, long → Print values → End

C Code

```c
#include <stdio.h>

int main() {
    printf("Size of int: %lu bytes\n", sizeof(int));
    printf("Size of float: %lu bytes\n", sizeof(float));
    printf("Size of char: %lu bytes\n", sizeof(char));
    printf("Size of double: %lu bytes\n", sizeof(double));
    printf("Size of short: %lu bytes\n", sizeof(short));
    printf("Size of long: %lu bytes\n", sizeof(long));

    return 0;
}
```

---

4)Bitwise AND, OR, XOR Operations

Algorithm

1. Take two unsigned int inputs from the user.

2. Perform AND (&), OR (|), XOR (^) operations.

3. Print the results.

Flowchart

Start → Input two unsigned int numbers → Perform AND, OR, XOR → Print results → End

C Code

```
#include <stdio.h>

int main() {
    unsigned int a, b;

    printf("Enter two unsigned integers: ");
    scanf("%u %u", &a, &b);

    printf("AND: %u\n", a & b);
    printf("OR: %u\n", a | b);
    printf("XOR: %u\n", a ^ b);

    return 0;
}
```

---

5)Basic Arithmetic Operations

Algorithm

1. Take two integers as input.

2. Calculate sum, difference, product, and quotient.

3. Print the results.

Flowchart

Start → Input two integers → Compute sum, difference, product, quotient → Print results → End

C Code

```
#include <stdio.h>

int main() {
    int a, b;
```

```
    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);

    printf("Sum: %d\n", a + b);
    printf("Difference: %d\n", a - b);
    printf("Product: %d\n", a * b);

    if (b != 0)
        printf("Quotient: %.2f\n", (float)a / b);
    else
        printf("Division by zero is not allowed.\n");

    return 0;
}
```

---

6)ASCII Value of a Character

Algorithm

1. Take a character input.

2. Print its ASCII value.

Flowchart

Start → Input character → Print ASCII value → End

C Code

```
#include <stdio.h>

int main() {
    char ch;

    printf("Enter a character: ");
    scanf(" %c", &ch);

    printf("ASCII value of %c is %d\n", ch, ch);

    return 0;
}
```

---

7)Print Float with Two Decimal Places

Algorithm

1. Take a float input.

2. Print it with 2 decimal places.

Flowchart

Start → Input float → Print with 2 decimal places → End

C Code

```
#include <stdio.h>

int main() {
    float num;

    printf("Enter a floating-point number: ");
    scanf("%f", &num);

    printf("Formatted number: %.2f\n", num);

    return 0;
}
```

---

8. Swap Two Integers Without a Third Variable

Algorithm

1. Take two integers as input.

2. Swap using arithmetic operations.

3. Print the swapped values.

Flowchart

Start → Input two integers → Swap using arithmetic → Print swapped values → End

C Code

```
#include <stdio.h>

int main() {
    int a, b;

    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);

    a = a + b;
    b = a - b;
```

```
    a = a - b;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

---

9)Check Even or Odd

Algorithm

1. Take an integer input.

2. Use modulus (%) to check divisibility by 2.

3. Print "Even" or "Odd".

Flowchart

Start → Input integer → Check num % 2 → Print Even/Odd → End

C Code

```
#include <stdio.h>

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    if (num % 2 == 0)
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

OPERATORS :

1. Swap Two Numbers Without Using a Temporary Variable

Algorithm

1. Take two integers as input.

2. Swap using arithmetic operations.

3. Print the swapped values.

Flowchart

Start → Input two integers → Swap using arithmetic → Print swapped values → End

C Code

```
#include <stdio.h>

int main() {
    int a, b;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    a = a + b;
    b = a - b;
    a = a - b;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

---

2)Check Even or Odd Using Bitwise Operators

Algorithm

1. Take an integer input.

2. Use bitwise AND (&) with 1 to check if it's even or odd.

3. Print the result.

Flowchart

Start → Input integer → num & 1 → Print Even/Odd → End

C Code

```
#include <stdio.h>

int main() {
    int num;

    printf("Enter an integer: ");
```

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

    if (num & 1)
        printf("Odd\n");
    else
        printf("Even\n");

    return 0;
}
```

---

3. Bitwise AND, OR, XOR Operations

Algorithm

1. Take two integers as input.

2. Perform AND (&), OR (|), XOR (^) operations.

3. Print the results.

Flowchart

Start → Input two integers → Perform AND, OR, XOR → Print results → End

C Code

```
#include <stdio.h>

int main() {
    int a, b;

    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);

    printf("AND: %d\n", a & b);
    printf("OR: %d\n", a | b);
    printf("XOR: %d\n", a ^ b);

    return 0;
}
```

---

4)Count the Number of Set Bits (1s) in Binary Representation

Algorithm

1. Take an integer as input.

2. Use a loop to count the number of 1s in the binary representation.

3. Print the count.

Flowchart

Start → Input integer → Loop: count set bits → Print count → End

C Code

```
#include <stdio.h>

int countSetBits(int num) {
    int count = 0;
    while (num) {
        count += num & 1;
        num >>= 1;
    }
    return count;
}

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    printf("Number of set bits: %d\n", countSetBits(num));

    return 0;
}
```

---

5)Check if a Number is a Power of Two

Algorithm

1. Take an integer as input.

2. Use num & (num - 1) == 0 condition to check if it's a power of two.

3. Print the result.

Flowchart

Start → Input integer → Check (num & (num - 1) == 0) → Print result → End

C Code

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

```
int isPowerOfTwo(int num) {
    return num > 0 && (num & (num - 1)) == 0;
}

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    if (isPowerOfTwo(num))
        printf("%d is a power of two.\n", num);
    else
        printf("%d is not a power of two.\n", num);

    return 0;
}
```

---

6)Check if a Number is Positive, Negative, or Zero Using Conditional Operators

Algorithm

1. Take an integer input.

2. Use conditional operators (? :) to check its sign.

3. Print the result.

Flowchart

Start → Input integer → Check sign using ? : → Print result → End

C Code

```
#include <stdio.h>

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    (num > 0) ? printf("Positive\n") : (num < 0 ? printf("Negative\n") : printf("Zero\n"));

    return 0;
}
```

---

7)Check if a Number is a Strong Number

Algorithm

1. Take an integer as input.

2. Compute the sum of the factorials of its digits.

3. Compare with the original number.

4. Print if it's a Strong number.

Flowchart

Start → Input integer → Compute sum of factorials of digits → Compare with original number → Print result → End

C Code

```
#include <stdio.h>

int factorial(int n) {
    int fact = 1;
    for (int i = 1; i <= n; i++)
        fact *= i;
    return fact;
}

int isStrong(int num) {
    int sum = 0, temp = num;
    while (temp) {
        sum += factorial(temp % 10);
        temp /= 10;
    }
    return sum == num;
}

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    if (isStrong(num))
        printf("%d is a Strong number.\n", num);
    else
        printf("%d is not a Strong number.\n", num);
```

```
    return 0;
}
```

---

8. Check if a Number is a Palindrome

Algorithm

1. Take an integer as input.

2. Reverse the number.

3. Compare it with the original.

4. Print if it's a palindrome.

Flowchart

Start → Input integer → Reverse number → Compare with original → Print result → End

C Code

```
#include <stdio.h>

int isPalindrome(int num) {
    int reversed = 0, temp = num;
    while (temp) {
        reversed = reversed * 10 + (temp % 10);
        temp /= 10;
    }
    return reversed == num;
}

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    if (isPalindrome(num))
        printf("%d is a Palindrome.\n", num);
    else
        printf("%d is not a Palindrome.\n", num);

    return 0;
}
```

Arrays :

1. Access an Array Element

Algorithm

1. Declare an array.

2. Access and print a specific element.

Flowchart

Start → Declare array → Access element → Print element → End

C Code

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};

    printf("The third element is: %d\n", arr[2]);

    return 0;
}
```

---

2. Change the Value of a Specific Element

Algorithm

1. Declare an array.

2. Modify a specific element.

3. Print updated array.

Flowchart

Start → Declare array → Modify element → Print updated array → End

C Code

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};

    arr[2] = 100; // Changing the third element

    printf("Updated array: ");
```

```
    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

---

3. Loop Through an Array

Algorithm

1. Declare an array.

2. Use a loop to print all elements.

Flowchart

Start → Declare array → Loop through array → Print elements → End

C Code

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};

    printf("Array elements: ");
    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

---

4. Create an Array and Add Elements Later

Algorithm

1. Declare an array with a specific size.

2. Take input values.

3. Print the array.

Flowchart

Start → Declare array → Input values → Print array → End

C Code

```c
#include <stdio.h>

int main() {
    int arr[5];

    printf("Enter 5 elements: ");
    for (int i = 0; i < 5; i++)
        scanf("%d", &arr[i]);

    printf("Array elements: ");
    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

---

5. Print the Size of the Array

Algorithm

1. Declare an array.

2. Use sizeof() to find its size.

3. Print the size.

Flowchart

Start → Declare array → Compute size using sizeof() → Print size → End

C Code

```c
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};

    printf("Size of array: %lu bytes\n", sizeof(arr));

    return 0;
}
```

---

6. Find the Length of an Array

Algorithm

1. Declare an array.

2. Compute its length using sizeof().

3. Print the length.

Flowchart

Start → Declare array → Compute length → Print length → End

C Code

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};

    int length = sizeof(arr) / sizeof(arr[0]);
    printf("Length of array: %d\n", length);

    return 0;
}
```

---

7. Calculate the Average Age

Algorithm

1. Declare an array.

2. Compute the sum of all elements.

3. Divide sum by the number of elements.

4. Print the average.

Flowchart

Start → Declare array → Compute sum → Compute average → Print result → End

C Code

```
#include <stdio.h>

int main() {
    int ages[] = {18, 22, 20, 25, 30};
    int sum = 0, count = sizeof(ages) / sizeof(ages[0]);

    for (int i = 0; i < count; i++)
        sum += ages[i];
```

```
    printf("Average age: %.2f\n", (float)sum / count);

    return 0;
}
```

---

8. Find the Lowest Age

Algorithm

1. Declare an array.

2. Initialize min with the first element.

3. Loop through the array to find the smallest value.

4. Print the result.

Flowchart

Start → Declare array → Initialize min → Loop through array → Print minimum value → End

C Code

```
#include <stdio.h>

int main() {
    int ages[] = {18, 22, 20, 25, 30};
    int min = ages[0];

    for (int i = 1; i < 5; i++)
        if (ages[i] < min)
            min = ages[i];

    printf("Lowest age: %d\n", min);

    return 0;
}
```

---9. Reverse the Elements of an Array

Algorithm

1. Declare an array.

2. Use a loop to swap elements from start to end.

3. Print the reversed array.

Flowchart

Start → Declare array → Swap elements → Print reversed array → End

C Code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int length = sizeof(arr) / sizeof(arr[0]);

    printf("Reversed array: ");
    for (int i = length - 1; i >= 0; i--)
        printf("%d ", arr[i]);

    return 0;
}
```

---

10. Find the Sum of All Elements in an Array

Algorithm

1. Declare an array.

2. Use a loop to compute the sum.

3. Print the sum.

Flowchart

Start → Declare array → Compute sum → Print sum → End

C Code

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int sum = 0, length = sizeof(arr) / sizeof(arr[0]);

    for (int i = 0; i < length; i++)
        sum += arr[i];

    printf("Sum of elements: %d\n", sum);

    return 0;
}
```
