Daffodil International University

Course Title: Structured Programming

Course Code: CIS 122

Semester Final

Fall 2024

By: Aadil

Table of Contents

☆ ☆ ☆ What are the features of the C language?

Answer:

1. **Reliability**: C helps create stable programs by offering good error handling and control over memory.
2. **Portability**: C programs can run on different computers with little or no changes because of its standard rules (ANSI/ISO C).
3. **Flexibility**: C allows working closely with memory, creating custom data types, and using special instructions to modify how the program is built.
4. **Modularity**: C makes it easy to split a program into smaller parts (modules). Functions and header files organize the code, making it easier to manage and fix.
5. **Efficiency**: C is fast and uses memory well, making it a good choice for programs that need to run quickly or use limited resources.

☆ What are the uses of C language?

Answer:

1. **System Programming**: Used to develop **operating systems** and **device drivers**.
2. **Embedded Systems**: Used in **microcontrollers**, **robots**, and **IoT devices**.
3. **Compilers and Interpreters**: Many **compilers** and **programming tools** are written in C.
4. **Game Development**: Used to create **games** and **game engines** for high performance.
5. **Networking**: Used in developing **networking software** and **communication protocols**.
6. **Scientific Applications**: Used for **scientific computing** and **engineering simulations**.
7. **Database Systems**: C is used to build **databases** like **MySQL**.
8. **Real-Time Systems**: Used in **real-time systems** like **medical devices** and **flight control**.

☆☆☆ Why is C called a mid-level language?

Answer:

C is called a **mid-level language** because it has features of both **low-level** and **high-level** languages:

1. **Low-Level Features:**C allows **direct access to memory** using pointers, just like assembly language. This makes it good for **system programming** like writing operating systems.
2. **High-Level Features:**C also has **functions**, **loops**, and **data types** (like int, float) which are found in high-level languages. These features make it easier to write complex programs.

So, **C** is considered a mid-level language because it combines the power of **low-level control** with the ease of **high-level programming**.

☆☆☆ What is Structured Programming? What are the differences between Structured Programming (SP) and Object-Oriented Programming (OOP)?

Answer:

Structured Programming is about improving code organization and readability by focusing on logical control structures and breaking problems into smaller functions.

Differences between SP and OOP:

Focus:

- **SP:** Focuses on functions and how the program flows.
- **OOP:** Focuses on objects and how they interact.

Data and Functions:

- **SP:** Data and functions are separate.
- **OOP:** Data and functions are combined in objects.

Maintenance:

- **SP:** Hard to maintain as programs grow.
- **OOP:** Easier to maintain and extend.

Languages:

- **SP:** C, Pascal, Fortran.
- **OOP:** Java, Python, C++, C#.

☆☆ What is a Variable, and What is the Correct Way to Write a Variable Name?

Answer:

A variable is a container used to store data that can change during the execution of a program.

Correct Way to Write a Variable Name:

1. **Starts with a letter or underscore:**
    - **Correct:** name, _name, userAge
    - **Incorrect:** 123name, 2ndNumber
2. **Can include letters, numbers, and underscores:**
    - **Correct:** user_age, totalAmount3
    - **Incorrect:** user@age
3. **No spaces allowed:**
    - **Correct:** firstName, total_amount
    - **Incorrect:** first name

4. **Cannot use reserved keywords:**
    - **Incorrect:** if, class, for
5. **Case-sensitive:**
    - **Correct:** age, Age
    - **Incorrect:** age and AGE are treated as different variables.
6. **Choose meaningful names:**
    - **Good practice:** userAge, totalPrice
    - **Bad practice:** x, temp

☆ How do you declare and assign a variable in C?

Answer:

```
#include <stdio.h>
int main() {
   int age = 25;     // Declare an integer variable and assign a value
   printf("The age is: %d\n", age);      // Print the value of the variable
   return 0;
}
```

☆☆ What are the different types of tokens in C programming? Provide examples for each type.

Answer:

1. **Keywords**: Reserved words with special meaning.
   Example: `int`, `if`, `return`.
2. **Identifiers**: Names given to variables or functions.
   Example: `age`, `totalAmount`.
3. **Constants**: Fixed values that do not change.
   Example: `10`, `3.14`, `'A'`.
4. **String** : Sequences of characters in double quotes.
   Example: `"Hello"`, `"World"`.
5. **Operators**: Symbols that perform operations on variables.
   Example: `+`, `-`, `*`, `==`.
6. **Punctuation**: Symbols that organize or separate code.
   Example: `;`, `,`, `()`, `{}`, `[]`.

☆☆☆ Explain the basic structure of a C program with an example.

Answer:

```
#include <stdio.h>  // Preprocessor Directive
int main() {  // Main Function
   int x = 5, y = 10, result;  // Declare variables
   result = x + y;  // Add x and y
   printf("Sum: %d\n", result);  // Print result
   return 0;  // End program
}
```

Explanation:

1. **Preprocessor Directive**: `#include <stdio.h>` includes the standard I/O functions like `printf`.
2. **Main Function**: The program starts here.
    - Declares variables `x`, `y`, and `result`.
    - Adds `x` and `y`, storing the result in `result`.
    - Prints the sum using `printf`.
3. **Return Statement**: `return 0;` ends the program, indicating successful execution.

☆☆☆ What is Pseudocode, Algorithm, and Flowchart?

Answer:

- **Pseudocode** is a simple way of describing the steps of a solution in plain language, using a mix of regular words and some programming-style instructions.
- **Algorithm** is a clear, step-by-step list of instructions to solve a problem.
- **Flowchart** is a picture or diagram that shows the steps of a process using shapes like rectangles for actions and diamonds for decisions.

☆ What is recursion in programming?

Answer:

Recursion is when a function calls itself to solve a problem. The function keeps calling itself with smaller parts of the problem until it reaches a simple case.

☆☆☆ What is the difference between Pseudocode, Algorithm, and Flowchart?

Answer:

Aspect	Pseudocode	Algorithm	Flowchart
Definition	A human-readable description of steps.	A step-by-step procedure to solve a task.	A graphical representation of steps.
Format	Plain language, mix of code-like syntax.	Structured, step-by-step instructions.	Uses symbols (e.g., rectangles, diamonds).
Purpose	To plan logic before coding.	To define the solution method.	To visualize the algorithm.
Detail Level	Less formal, simple structure.	Precise, detailed steps.	Visual with specific symbols for actions.
Ease of Understanding	Easy to understand for humans.	Requires logical understanding.	Easy to follow visually.
Examples	`Start`, `Set x = 5`, `If x > 0 then`	1. Start 2. Set x = 5 3. If x > 0	Start → Set x = 5 → If x > 0 → Print "Positive"

☆☆☆ Write a pseudocode to read the two sides of a rectangle and calculate its area.

Answer:

```
START
   Declare f bat length, width, area
   // Step 2: Read the length and width of the rectangle from the user
   PRINT "Enter the length of the rectangle: "
   READ length
   PRINT "Enter the width of the rectangle: "
   READ width
   // Step 3: Calculate the area of the rectangle
   area = length * width
   // Step 4: Display the result
```

PRINT "The area of the rectangle is: ", area

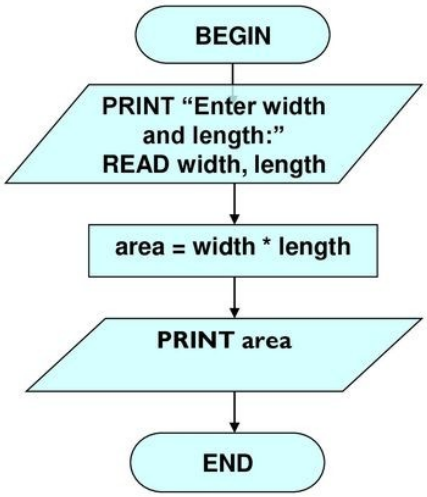
END

Example: Write down an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.

Algorithm:

1. BEGIN
2. PRINT "Enter width and length:"
3. READ width, length
4. area = width * length
5. PRINT area
6. END

Flowchart



10

☆☆☆ Explain the do-while loop in C with an example.

Answer:

```
#include <stdio.h>
int main() {
   int i = 1;
   do {
      printf("Value of i: %d\n", i);
      i++;  // Increment i by 1
   } while (i <= 5);  // Loop runs while i is less than or equal to 5
   return 0; }
```

Explanation:

1. **Initialization**: The variable `i` is initialized to `1`.

2. **First Execution**: The `do` block is executed, and it prints `i = 1`.
3. **Condition Check**: After executing the loop body, the condition `i <= 5` is checked.
4. **Loop Continuation**: Since `i` is still less than or equal to 5, the loop runs again, and `i` is incremented.
5. **Termination**: When `i` becomes 6, the condition `i <= 5` becomes false, and the loop terminates.

☆☆☆ What is an array, and why is it more efficient than using variables?

Answer:

An array is a data structure that allows you to store multiple values of the same type in a single variable.

Why Arrays are More Efficient Than Individual Variables:

1. **Memory**: Arrays use a single block of memory, while variables use separate memory for each element.
2. **Speed**: Arrays improve CPU cache usage, making access faster.
3. **Simplicity**: You can loop through an array, while handling multiple variables requires repetitive code.
4. **Dynamic Size**: Arrays can be dynamically sized at runtime, unlike individual variables, which have a fixed size.

☆☆☆ What is a function, and why is it important in programming?

Answer:

A function is a block of code designed to perform a specific task. It can take inputs (parameters), perform an action, and optionally return a result.

Why Functions Are Important:

1. **Code Reusability**: Write the code once and use it multiple times.
2. **Modularity**: Break down complex tasks into smaller, manageable parts.
3. **Abstraction**: Hide complex details, making code easier to understand.
4. **Easier Debugging**: Functions are self-contained, so they are easier to test and debug.
5. **Organization**: Functions help organize code logically.

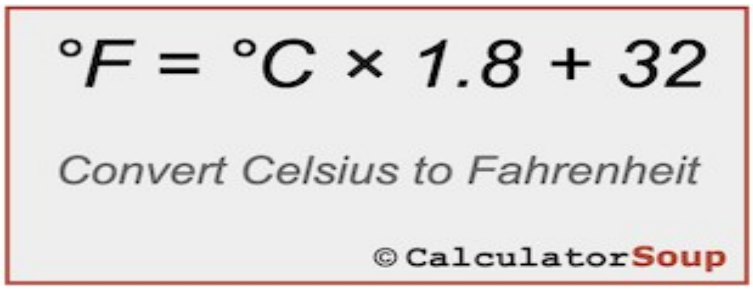
☆☆☆ Write a C program that takes a number between 1 and 5 as input and prints its spelling (in words) using a `switch` statement.

Answer:

```
#include <stdio.h>
int main() {
   int number;
   scanf("%d", &number);
   switch (number) {
      case 1: printf("One\n"); break;
      case 2: printf("Two\n"); break;
      case 3: printf("Three\n"); break;
      case 4: printf("Four\n"); break;
      case 5: printf("Five\n"); break;
      default: printf("Invalid\n");
   }
   return 0;
}
```

☆☆☆ Write a C program that converts temperature from Fahrenheit to Celsius.

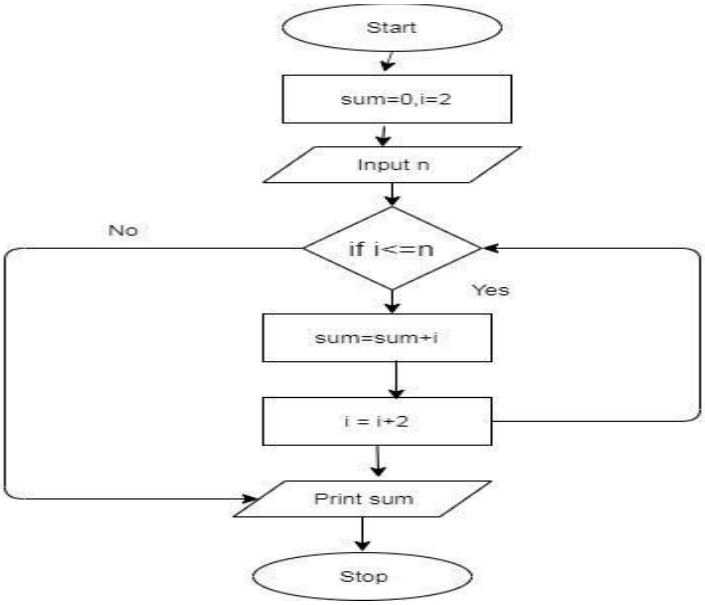
Answer:




```
#include <stdio.h>
int main() {
   float fahrenheit, intermediate, celsius;
   scanf("%f", &fahrenheit);
   intermediate = (fahrenheit - 32) * 5;
   celsius = intermediate / 9;
   printf("%.2f Fahrenheit = %.2f Celsius\n", fahrenheit, celsius);
   return 0;
}
```

☆☆☆ Write a C program to calculate the sum of the series: 2 + 4 + 6 + ... up to n terms, where n is provided by the user. Also write the flowchart.

Answer:




```
#include <stdio.h>
int main() {
   int n, i, sum = 0;
   printf("Enter the last number of the series: ");
   scanf("%d", &n);

   for (i = 2; i <= n; i = i + 2) {
      sum = sum + i;
   }
   printf("Sum of the series: %d\n", sum);
   return 0;
}
```

☆☆☆ Write a C program to calculate the sum of the series: $1^2 + 2^2 + 3^2 + ... + n^2$, where n is provided by the user.

Answer:

```
#include <stdio.h>
int main() {
   int n, sum = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);

   for (int i = 1; i <= n; i++) {
      sum =sum +( i * i);
   }
   printf("Sum of squares: %d\n", sum);
   return 0;
}
```

☆☆☆ Write a C program to print a pyramid pattern as shown below. The number of rows in the pyramid should be provided by the user.

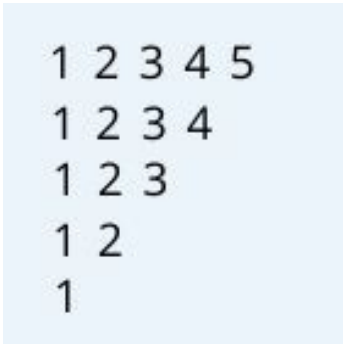
Answer:


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

   for (row = 1; row <= n; row++) {
      for (col = 1; col <= row; col++) {
         printf("%d ", col); // if you wanna print * then just replace it.
      }
      printf("\n");
   }
}
```

☆☆☆ Write a C program to print a pyramid pattern as shown below. The number of rows in the pyramid should be provided by the user.

Answer:




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

   for (row = n; row >= 1; row--) {
      for (col = 1; col <= row; col++) {
         printf(" *");
      }
      printf("\n");
   }
}
```

☆☆ Write a C program to calculate x raised to the power of y (i.e., xy), where x and y are provided by the user.

Answer:

```
#include <stdio.h>
#include <math.h>
int main() {
   int num1, num2;
   double result;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);

   result = pow(num1, num2);
   printf("Result: %.0lf\n", result);
}
```

☆☆☆ Write a C program to check if a given number is a prime number or not.

Answer:

```
#include <stdio.h>
int main() {
   int num, i;
   printf("Enter a number: ");
   scanf("%d", &num);
   if (num < 2) {
      printf("%d is not a prime number.\n", num);
   } else {
      for (i = 2; i <= num / 2; i++) {
         if (num % i == 0) {
            printf("%d is not a prime number.\n", num);
            return 0;
         }
      }
      printf("%d is a prime number.\n", num);
   }
}
```

☆☆☆ Write a C program to print all prime numbers from 1 to n, where n is provided by the user.

Answer:

```
#include <stdio.h>
int main() {
    int num, i, j, isPrime;
    printf("Enter the value of n: ");
    scanf("%d", &num);
    printf("Prime numbers from 1 to %d are:\n", num);

    for (i = 2; i <= num; i++) {
        isPrime = 1;
        for (j = 2; j <= i / 2; j++) {
            if (i % j == 0) {
                isPrime = 0;
                break;
            }
        }
        if (isPrime) {
            printf("%d ", i);
        }
    }
    printf("\n");
}
```

☆☆☆ Write a C program to print the first n Fibonacci numbers, where n is provided by the user.

Answer:

```
#include <stdio.h>
int main() {
   int n, a = 0, b = 1, next, count = 0;
   printf("Enter the terms in the Fibonacci sequence (e.g., 10): ");
   scanf("%d", &n);

   for (count = 0; count < n; count++) {
      printf("%d ", a);
      next = a + b;
      a = b;
      b = next;
   }
   printf("\n");
}
```

☆☆☆ Write a C program to calculate the factorial of a given number .

Answer:

```
#include <stdio.h>
int main() {
   int num, i;
   long long factorial = 1;
   printf("Enter a number: ");
   scanf("%d", &num);

   for (i = 1; i <= num; i++) {
      factorial = factorial * i;
   }
   printf("The factorial of %d is: %lld\n", num, factorial);

}
```

☆☆☆ Write a C program to read a number from the user and calculate the sum of those digits.

Answer:

```
#include <stdio.h>
int main() {
   int num, sum = 0;
   printf("Enter a number: ");
   scanf("%d", &num);

   while (num > 0) {
      sum  = sum + (num % 10);
      num /= 10;
   }
   printf("\nSum of its digits is: %d\n", sum);
}
```

☆☆Write a C program that reverses the digits of a number entered by the user.

Answer:

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

   char command[50];
   sprintf(command, "echo %d | rev", n);
   system(command);
}
```

☆☆☆ Write a C program to calculate the Greatest Common Divisor (GCD) and Least Common Multiple (LCM) of two integers.

Answer:

```
#include <stdio.h>
int main() {
   int num1, num2, gcd, lcm, a, b;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);
   a = num1;
   b = num2;

   while (b != 0) {
      int temp = b;
      b = a % b;
      a = temp;
   }
   gcd = a;
   lcm = (num1 * num2) / gcd;
   printf("GCD = %d\n", gcd);
   printf("LCM = %d\n", lcm);
}
```

☆☆☆ Write a C program that accepts an array of integers from the user and then calculates and displays the sum of the array elements.

Answer:

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   int array[n];
   printf("Enter %d elements of the array:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &array[i]);
   }
   int sum = 0;
   for (int i = 0; i < n; i++) {
      sum =sum +  array[i];
   } printf("Sum of the array = %d\n", sum);
}
```

☆☆☆ Write a C Program to Find and Display the Highest Integer in the Array.

Answer:

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   int array[n];
   printf("Enter %d elements of the array:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &array[i]);
   }
   int highest = array[0];
   for (int i = 1; i < n; i++) {
      if (array[i] > highest) {
         highest = array[i];
      }
   }printf("The highest number in the array is: %d\n", highest);
}
```

☆☆☆ Write a C Program to Insert a Number in an Array.

Answer:

```
#include <stdio.h>
int main() {
   int n, pos, value;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   int array[n];

   printf("Enter %d elements of the array:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &array[i]);
   }
   printf("Enter the position to insert (1 to %d): ", n + 1);
   scanf("%d", &pos);
   printf("Enter the value to insert: ");
   scanf("%d", &value);

   for (int i = n - 1; i >= pos - 1; i--) {
      array[i + 1] = array[i];
   }
   array[pos - 1] = value;
   n++;

   printf("Array after insertion:\n");
   for (int i = 0; i < n; i++) {
      printf("%d ", array[i]);
   }
}
```

☆☆ Write a C Program to update a Number in an Array.

Answer:

```
#include <stdio.h>
int main() {
   int n, pos, value;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   int array[n];

   printf("Enter %d elements of the array:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &array[i]);
   }
   printf("Enter the position to update (1 to %d): ", n);
   scanf("%d", &pos);
   printf("Enter the value to insert: ");
   scanf("%d", &value);

   array[pos - 1] = value;

   printf("Array after update:\n");
   for (int i = 0; i < n; i++) {
      printf("%d ", array[i]);
   }
}
```

☆☆☆ Write a C Program to delete a Number in an Array.

Answer:

```
#include <stdio.h>
int main() {
   int n, pos;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   int array[n];

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

   printf("Enter the position to delete (1 to %d): ", n);
   scanf("%d", &pos);

   for (int i = pos - 1; i < n - 1; i++) {
      array[i] = array[i + 1];
   }
   n--;

   printf("Array after delete:\n");
   for (int i = 0; i < n; i++) {
      printf("%d ", array[i]);
   }
}
```

☆☆☆ Write a C Program to search a Number in an Array.

Answer:

```
#include <stdio.h>
int main() {
   int n, s, found = 0;
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);
   int array[n];

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

   printf("Enter the number to search: ");
   scanf("%d", &s);

   for (int i = 0; i < n; i++) {
      if (array[i] == s) {
         printf("Found at position %d\n", i + 1);
         found = 1;
         break;
      }
   }
   if (!found) {
      printf("Not found\n");
   }
}
```

☆☆☆ Loop conversion between For, While, and Do-while loops.

Answer:

*** Example of a** for **loop:**

```
for (int k = 1; k <= 10; k++) {
   printf("%d\n", k);
}
```

*** Example of a while loop:**

```
int k = 1;
while (k <= 10) {
   printf("%d\n", k);
   k++;
}
```

***Example of a do-while loop:**

```
int k = 1;
do {
   printf("%d\n", k);
   k++;
} while (k <= 10);
```

Note: Maybe I skipped some important things, so further investigate your slides

- LunarLumos