C++ Help Book By Coolboymannu

B.A (Prog) with Computer Science as Major/Minor
DSC01: Introduction to Programming using C++

S. No.	Unit Name	Chapters	References	Weeks
1.	Unit 1 Introduction to C++	1 (upto page no 22)	[2]	1
		2	[1]	
2.	Unit 2 Data types and Expressions	2	[2]	2 – 4
3.	Unit 3 Control Constructs in C++	3	[2]	5 – 8
4.	Unit 4 Arrays, Pointers and User Defined Functions	5 (162 – 171, 176 – 178, 182 – 186, 188 – 193, 195 – 199, 206 – 207), 7 (upto page no 276), 10 (upto page no 438)	[2]	9 – 10
5.	Unit 5 Classes and Objects	6 (upto page no 243)	[2]	11 – 15
		8 (8.1 - 8.7)	[1]	

Essential Readings
1. E. Balaguruswamy, Object Oriented Programming with C++, 7th edition, McGraw – Hill Education, 2017.
2. Robert Lafore, Object Oriented Programming in C++, 4th edition, SAMS Publishing, 2016.

Certainly! Below is an enhanced and more understandable version of the syllabus for DSC-1 (Programming Fundamentals Using C++):

Unit 1: Introduction to C++ (3 hours)

- Understanding Object-Oriented Programming (OOP)
- Characteristics of OOP
- Structure of a C++ Program:
  - main() function
  - Header files
  - Output and Input
  - Comments
- Compiling and executing a simple program

Unit 2: Data Types and Expressions (9 hours)

- Keywords and their significance
- Built-in data types
- Variables and constants
- Naming conventions
- Input-Output statements
- Operators and their precedence
- Expressions
- Typecasting
- Library functions

Unit 3: Control Constructs in C++ (12 hours)

- Decision making using selection constructs
- Iteration using looping constructs

Unit 4: Arrays, Pointers, and User-Defined Functions (6 hours)

- Defining and initializing single and multi-dimensional arrays
- User-defined functions
- Passing arguments to functions
- Returning values from functions
- Inline functions
- Default arguments
- Introduction to pointers

Unit 5: Classes and Objects (15 hours)

- Need for abstraction, encapsulation, inheritance, and polymorphism
- Creating classes
- Objects as function arguments
- Modifiers and access control
- Constructors and destructors

This organised and detailed breakdown should provide a comprehensive structure for your DSC-1 theory course. Feel free to use this as a foundation for your videos, ensuring you delve into each topic with examples and practical demonstrations.

Short Questions and answers

Questions

1. **What are the characteristics of Object-Oriented Programming (OOP)?**
2. **Define variables and constants in C++.**
3. **Explain the concept of naming conventions for variables in C++.**
4. **What is the significance of operators and their precedence in C++?**
5. **What are the keywords in C++?**
6. **List and explain the different types of data types in C++**
7. **Explain the purpose of typecasting in C++ and provide a scenario where it is necessary.**
8. **Demonstrate the implementation of decision-making using the switch statement in C++.**
9. **What is the significance of iteration in programming?**
10. **What are the different types of loops in C++ for iteration?**
11. **Discuss the concept of default arguments in C++ functions. Provide an example.**
12. **Define arrays in C++.**
13. **Discuss the importance of user-defined functions in C++ and their advantages.**
14. **Explain the role of pointers in C++.**
15. **Why is abstraction important in programming?**
16. **Explain the need for abstraction, encapsulation, inheritance, and polymorphism in C++ classes.**
17. **Explain the principles of encapsulation and how it enhances the security of a C++ program.**
18. **Explain the implementation of inheritance in C++.**
19. **What is polymorphism, and how is it achieved in C++?**
20. **What is the role of constructors and destructors in C++?**

Answers

1. **Characteristics of Object-Oriented Programming (OOP):**
    - Object-Oriented Programming is characterized by four main principles: encapsulation, inheritance, polymorphism, and abstraction. These principles facilitate better code organization, reusability, and maintenance.

2. **Variables and Constants in C++:**
    - Variables in C++ are containers that store data, and their values can change during program execution. Constants, on the other hand, have fixed values that do not change during runtime.
3. **Naming Conventions for Variables in C++:**
    - Naming conventions in C++ define rules for naming variables to enhance code readability. Common practices include using camelCase or underscores to separate words in variable names.
4. **Significance of Operators and Precedence in C++:**
    - Operators in C++ perform operations on variables and values. Precedence determines the order in which these operations are executed. Understanding operator precedence is crucial for writing accurate expressions.
5. **Keywords in C++:**
    - Keywords in C++ are reserved words with predefined meanings. Examples include `int`, `if`, `else`, and `switch`.
6. **Types of Data Types in C++:**
    - C++ has built-in data types such as int, float, char, and double. User-defined data types include structures and classes.
7. **Purpose of Typecasting in C++:**
    - Typecasting is used to convert data from one type to another. For example, converting an integer to a floating-point number is necessary when performing certain arithmetic operations.
8. **Implementation of Switch Statement in C++:**
    - The switch statement in C++ is used for decision-making. It allows the program to execute different code blocks based on the value of a variable.
9. **Significance of Iteration in Programming:**
    - Iteration in programming allows executing a block of code repeatedly. It is fundamental for tasks that require repetitive execution, such as processing arrays or lists.
10. **Types of Loops in C++ for Iteration:**
    - C++ supports various loops for iteration, including the `for` loop, `while` loop, and `do-while` loop.
11. **Default Arguments in C++ Functions:**
    - Default arguments in C++ functions are values assigned to parameters if no arguments are provided during the function call. They enhance the flexibility of function calls.

12. **Arrays in C++:**
    - Arrays in C++ are collections of elements of the same data type. They provide a way to store and access multiple values using a single variable name.
13. **Importance of User-Defined Functions in C++:**
    - User-defined functions enhance code modularity and reusability by allowing the organization of code into separate, manageable functions.
14. **Role of Pointers in C++:**
    - Pointers in C++ store memory addresses. They are essential for dynamic memory allocation, efficient array manipulation, and working with functions.
15. **Abstraction in Programming:**
    - Abstraction involves simplifying complex systems by focusing on essential features and ignoring unnecessary details. It promotes clarity and modularity in code.
16. **Need for Abstraction, Encapsulation, Inheritance, and Polymorphism in C++ Classes:**
    - Abstraction, encapsulation, inheritance, and polymorphism are key principles in OOP that contribute to code organization, security, and flexibility within classes.
17. **Encapsulation Principles and Security Enhancement in C++:**
    - Encapsulation involves bundling data and methods within a class. It enhances security by controlling access to data through encapsulation.
18. **Implementation of Inheritance in C++:**
    - Inheritance in C++ allows a class to inherit properties and behaviors from another class. It facilitates code reuse and promotes a hierarchical structure.
19. **Polymorphism in C++:**
    - Polymorphism allows objects of different types to be treated as objects of a common type. It is achieved through function overloading and overriding.
20. **Role of Constructors and Destructors in C++:**
    - Constructors initialize objects, setting their initial state, while destructors clean up resources when an object goes out of scope or is explicitly deleted.

1. Find the largest of n natural numbers:

```cpp
#include <iostream>
using namespace std;

int main() {
    int n, num, largest = 0;

    cout << "Enter the value of n: ";
    cin >> n;

    cout << "Enter " << n << " natural numbers:\n";

    for (int i = 0; i < n; i++) {
        cin >> num;
        if (num > largest) {
            largest = num;
        }
    }

    cout << "The largest number is: " << largest << endl;

    return 0;
}
```

2. Check whether a given number is prime or not:

```cpp
#include <iostream>
using namespace std;

bool isPrime(int num) {
    if (num <= 1) return false;
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) return false;
    }
    return true;
}

int main() {
    int num;

    cout << "Enter a number: ";
    cin >> num;

    if (isPrime(num)) {
        cout << num << " is a prime number.\\n";
    } else {
        cout << num << " is not a prime number.\\n";
    }

    return 0;
}
```

3. Print a pattern of stars:

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;

    cout << "Enter the value of n: ";
    cin >> n;

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            cout << "* ";
        }
        cout << endl;
    }

    return 0;
}
```

4. Write a menu driven program for following:
    a. to check whether a given number is odd or even.
    b. display a Fibonacci series
    c. compute factorial of a number

```cpp
#include <iostream>
using namespace std;

double factorial(double n) {
    if (n == 0 || n == 1) return 1;
    return n * factorial(n - 1);
}
int main() {
    int choice;
    long num;
    cout << "Menu:\n";
    cout << "1. Check whether a number is odd or even.\n";
    cout << "2. Display Fibonacci series.\n";
    cout << "3. Compute factorial of a number.\n";
    cout << "Enter your choice (1-3): ";
    cin >> choice;
   int n, a = 0, b = 1, c;
    switch (choice) {
        case 1:
            cout << "Enter a number: ";
            cin >> num;
            if (num % 2 == 0){
                cout << num << " is even.\n";}
            else{
                cout << num << " is odd.\n";}
            break;
        case 2:
            cout << "Enter the number of terms in Fibonacci series: ";
            cin >> n;
            cout << "Fibonacci Series: ";
            for (int i = 1; i <= n; i++) {
                cout << a << " ";
                c = a + b;
                a = b;
                b = c;
            }
            cout << endl;
            break;
        case 3:
            cout << "Enter a number: ";
            cin >> num;
            cout << "Factorial of " << num << " is: " << factorial(num) << endl;
            break;
        default:
            cout << "Invalid choice.\n";
            break;
    }
    return 0;
}
```

5. Write a program to perform the following operations on an input string
    a. Print length of the string
    b. Find frequency of a character in the string
    c. Print whether characters are in uppercase or lowercase
    d. to check whether a given string is palindrome or not.

```cpp
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char str[100] , ch;
    int choice, length , frequency = 0;
    bool isPalindrome = true;
    cout << "Enter a string: ";
    cin >> str;
    length = strlen(str);
    cout << "1. Print length of the string.\n";
    cout << "2. Find frequency of a character in the string.\n";
    cout << "3. Print whether characters are in uppercase or lowercase.\n";
    cout << "4. Check whether the string is palindrome or not.\n";
    cout << "Enter your choice (1-4): ";cin >> choice;
    switch (choice) {
        case 1:
            cout << "Length of the string: " << length << endl;
            break;
        case 2:
            cout << "Enter a character to find its frequency: ";
            cin >> ch;
            for (int i = 0; i < length; i++) {
                if (str[i] == ch) frequency++;
            }
            cout << "Frequency of " << ch << ": " << frequency << endl;
            break;
        case 3:
            for (int i = 0; i < length; i++) {
                if (isupper(str[i]))
                    cout << str[i] << " is in uppercase.\n";
                else if (islower(str[i]))
                    cout << str[i] << " is in lowercase.\n";
            }
            break;
        case 4:
            for (int i = 0; i < length / 2; i++) {
                if (str[i] != str[length - i - 1]) {
                    isPalindrome = false;
                    break;}
            }
            if (isPalindrome)
                cout << "The string is a palindrome.\n";
            else
                cout << "The string is not a palindrome.\n";
            break;
        default:
            cout << "Invalid choice.\n";
            break;
    }
    return 0;}
```

1. Write a program to find the largest of n natural numbers.

2. Write a program to find whether a given number is prime or not.

3. Write a program that takes a positive integer n and the produce n lines of output as shown:

   *

   **

   ***

   ****

4. Write a menu driven program for following: a. to check whether a given number is odd or even. b. display a Fibonacci series c. compute factorial of a number
5. Write a program to accept a number, reverse it and print the sum of its digits.

6. Write a program using functions to print the series and its sum: 1 + 1/2! + 1/3! + … + 1/n!

7. Write a program to perform the following operations on an input string a. Print length of the string b. Find frequency of a character in the string c. Print whether characters are in uppercase or lowercase d. to check whether a given string is palindrome or not.

8. Write a program that will prompt the user for a list of 5 prices. Compute the average of the prices and find out all the prices that are higher than the calculated average.

9. Design a class named Vehicle, having registration number and year as its private members. Define a suitable constructor and a method to print the details of a vehicle. Write a C++ program to test the above class.

10. Inherit a class Car from the Vehicle class defined above. Add model to the Car class. Define a suitable constructor and a method to print the details of a car. Write a C++ program to test inheritance of this class.

C++ Programming Language Overview

Object-Oriented Programming (OOP)

Object-Oriented Programming is a paradigm that uses objects and classes to model real-world entities and their interactions. Key characteristics of OOP include:

- Abstraction: Hiding implementation details, exposing only essential features to the outside world.
- Encapsulation: Bundling data and functions within an object.
- Inheritance: Mechanism allowing a new class to be derived from an existing class, inheriting properties and behaviors.
- Polymorphism: Ability of an object to take on multiple forms, achieved through multiple classes or method overrides.

Structure of a Basic C++ Program

Elements of a basic C++ program include:

- Preprocessor Directives: Executed before compilation, includes header files, macros, and conditional statements.
- Main Function: Starting point of the program, executed first.
- Header Files: Contain declarations of functions and variables used in the program.
- Output: `cout` statement used to display output in the console.
- Input: `cin` statement used to get input from the user.
- Comments: Start with `//` to explain the program and enhance code readability.

Data Types in C++

Fundamental Data Types:

- `int` (integers)
- `float` (floating-point numbers)
- `double` (double-precision floating-point numbers)
- `char` (single characters)
- `bool` (true/false values)

Derived Data Types:

- Arrays
- Pointers
- References
- Structures
- Unions

- Classes

Variables and Constants

- Variables store values and can change during program execution.
- Constants are values that don't change during program execution.

Naming Convention for Variables and Constants

Descriptive names indicating what they represent.
No spaces in names.
Names shouldn't start with a number.
Avoid using keywords.

Operators and Their Precedence

Operators perform operations on values. Types include:

- Arithmetic Operators: `+, -, *, /, %`
- Relational Operators: `<, >, <=, >=, ==, !=`
- Logical Operators: `&&, ||, !`
- Assignment Operators: `=, +=, -=, *=, /=, %=`

Keywords in C++

Predefined words with special meanings (e.g., `int`, `float`, `if`, `while`, `class`).

Expressions in C++

Combinations of values, variables, and operators that can be evaluated (e.g., `2 + 3`).

Typecasting in C++

Converting a value from one data type to another using `(data type) expression`.

Library Functions in C++

Pre-written functions for common tasks (e.g., `cin`, `cout`, `sqrt`, `pow`).

Control Constructs in C++

- Selection Constructs:
  - `if` statement: Executes code if a condition is true.

    - `switch` statement: Executes code based on the value of an expression.
- Looping Constructs:
    - `while` loop: Repeats code while a condition is true.
    - `do-while` loop: Similar to `while`, but guaranteed to execute at least once.
    - `for` loop: Repeats code a specific number of times.

Arrays, User-Defined Functions, and Pointers in C++

- Arrays: Data structures storing a collection of values.
- User-Defined Functions: Blocks of code performing specific tasks.
- Pointers: Variables storing memory addresses.

Creating Classes and Objects in C++

- Abstraction and Encapsulation:
    - Abstraction achieved through classes, hiding internal details.
    - Encapsulation achieved by wrapping data and functions within a class.
- Inheritance:
    - Mechanism allowing a class to inherit properties and behaviors from another.
- Polymorphism:
    - Achieved through function overloading and operator overloading.

Strategy

1. What is your purpose? (getting good marks , placements , learning etc.)
2. understanding syllabus (you must know all topics in syllabus)
3. watching a full video of C++ or Python other video
4. solving PYQs
5. ready short answers and some important codes
6. self test
7. How to attempt paper
    1. take 1and half margin in paper
    2. there are 7 questions 1st is compulsory 2 to 7 answer only 4 questions
    3. look up questions serially (conscious mind)
    4. attempting questions depend on you

Best of luck