**Module-3**

1. What are the key differences between Procedural Programming and Object-Oriented Programming (OOP)?

Procedural Programming is Great for Straightforward task, while OOP helps manage complexity by breaking into modular objects.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Procedural Programming** | **Object-Oriented Programming (OOP)** |
| Approach | Focuses on functions & procedures to operate on data | Organizes code using objects & classes |
| Data Handling | Data is global, and functions act upon it | Encapsulates data within objects (data hiding) |
| Modularity | Functions are independent but not always reusable | Promotes reuse & modularity through inheritance & polymorphism |
| Security | Data is accessible directly, increasing risks | Uses encapsulation & access control to secure data |
| Scalability | Best suited for small programs with simpler workflows | Scales well for larger, complex applications |
| Examples | C, Pascal | C++, Java, Python (when using OOP concepts) |

1. List and explain the main advantages of OOP over POP.
2. Explain the steps involved in setting up a C++ development environment.
3. Search & Download Dev C++, Turbo C IDE from Google.
4. Install on System & create Icons on Desktop.
5. Open from icon and Create New file & Run it.
6. What are the main input/output operations in C++? Provide examples.

In C++, The main output and Input Operation is “cout and cin” using <iostream> library.

1. What are the different data types available in C++? Explain with examples.

Integer(int): - Store Whole Numbers

float/double: - Store decimal value

character(char): - Store a single character

Array: - Store Multiple Values of the same type.

Pointer: - Store Address of Multiple value

UDF: - Its help to create complex data types.

* Structure(struct): - Groups different Data types.
* Class: - Its collection of Data Member and Data Function.

1. Explain the difference between implicit and explicit type conversion in C++.

**Implicit Type Conversion**

Also known as automatic conversion or coercion, implicit conversion happens automatically by the compiler when different data types are mixed in an expression. Generally, it occurs when converting from a smaller data type (e.g., int) to a larger one (e.g., float) to prevent data loss.

**Explicit Type Conversion**

Also known as type casting, explicit conversion is performed manually by the programmer using casting operators. This is often needed when converting from a larger data type to a smaller one, or when the compiler cannot automatically perform the conversion.

1. What are the different types of operators in C++? Provide examples of each.

|  |  |
| --- | --- |
| **Operator** | **Examples** |
| **Assignment** | **"=, +=, -=, \*=, /=, %="** |
| **Arithmetic** | **"+ , - ,\* , / "** |
| **Relational** | **"==, !=, <, >, <=, >=, "** |
| **Logical** | **&&, || , !** |
| **increment/Decrement** | **"++, --"** |
| **Conditional** | **" ?, : "** |

1. Explain the purpose and use of constants and literals in C++.

Constant: -constant is a memory location whose value cannot be changed during program execution.

Literals:- It is direct Representation of fixed values within source code. They can be integers, float,character and string.

1. What are conditional statements in C++? Explain the if-else and switch statements.

The following are the conditional statements.

1. If else….
2. If else if
3. Nested if
4. Switch case

If else:- “When you have only one choice like either 1 or 2 then, we can use if else statement.

Switch case:-“When you have menu driven choice like ,There are number of choices and you have to select from them any one, use Switch Case Option.

1. What is the difference between for, while, and do-while loops in C++?

While and for loops are entry-controlled, meaning they check the condition before executing the loop body, while do-while is exit-controlled, checking the condition after the body is executed, guaranteeing at least one iteration.

1. How are break and continue statements used in loops? Provide examples.

In C++ language break and continue are jump statements used to control the flow of execution within loops and functions. break terminates the loop, continue skips the current iteration and proceeds to the next.

1. Explain nested control structures with an example.

A nested control structure, in programming, refers to placing one control structure (like if, else, for, while, etc.) inside another. This allows for complex logic by handling multiple conditions or iterating over multiple levels of data.

1. What is a function in C++? Explain the concept of function declaration, definition, and calling.

functions are reusable blocks of code that perform specific tasks. They enable modularity and code organization. A function declaration informs the compiler about the function's structure (return type, name, parameters), while the function definition provides the actual code that the function executes. A function is called by its name, potentially passing arguments, and it can return a value.

Types of Functions:

* Built in function ( cout, cin, etc…)
* User Define Function: -the function which are created by user for program are known as User define function.

|  |  |  |
| --- | --- | --- |
| Features | Local Space | Global space |
| Declaration | Inside a function or block | Outside any function or block |
| Accessibility | Only within the function or block | Anywhere in the program |
| Lifetime | Exists only while the function or block is executing | Exists for the entire duration of the program |

1. What is the scope of variables in C++? Differentiate between local and global scope.

In c++, the scope of variable determines where in the code that variable can be accessed. Variable in c++can have either local or global scope.

1. Explain recursion in C++ with an example.

Recursion in C++ is a programming technique where a function calls itself within its own definition.

This approach is useful for solving problems that can be broken down into smaller, self-similar subproblems. A recursive function typically has two parts: a base case and a recursive step.

The base case defines the condition under which the recursion stops, preventing an infinite loop. The recursive step calls the function itself with a modified input, moving closer to the base case.

1. What are function prototypes in C++? Why are they used?

A function prototype in C++ is a declaration of a function's name, return type, and parameters (including their types) before the actual function definition.

It informs the compiler about the function's existence and signature, enabling proper type checking and function call validation during compilation.

1. What are arrays in C++? Explain the difference between single-dimensional and multidimensional arrays.

Array is a data structure that stores a collection of elements of the same data type in contiguous memory locations. One-dimensional arrays store data in a linear sequence, while multi-dimensional arrays (like two-dimensional arrays) arrange data in a table or matrix format with rows and columns.

|  |  |  |
| --- | --- | --- |
| Key Differences: | | |
| **Feature** | **One-Dimensional Array** | **Two-Dimensional Array** |
| Structure | Single row or column | Grid or table (rows and columns) |
| Indices | Single index to access elements | Two indices (row and column) to access elements |
| Memory | Linear allocation of memory locations | Memory allocated contiguously, but organized in a tabular structure |
| Application | Storing lists, sequential data | Representing tables, matrices, spreadsheets |
| Example | List of names, a series of measurements | A spreadsheet of student scores, a chess board |

1. Explain string handling in C++ with examples.

String handling functions in C++ are used to manipulate and work with strings. These functions are defined in the <string> header.

1. Explain string operations and functions in C++.

Operations

* **Concatenation:** Strings can be joined using the + operator or the append () method.
* **Comparison:** Strings can be compared using relational operators (==, !=, <, >, <=, >=) or the compare () method.
* **Assignment:** Strings can be assigned values using the = operator.
* **Element Access:** Individual characters can be accessed using the [ ] operator or the at ( ) method.

**Functions**

* **length()/size():** Returns the number of characters in the string.
* **find():** Searches for a substring and returns its starting position.
* **substr():** Extracts a substring from a given position with a specified length.
* **replace():** Replaces a portion of the string with another string.
* **insert():** Inserts a string at a specific position.
* **erase():** Removes characters from the string.
* **clear():** Removes all characters from the string.

1. Explain the key concepts of Object-Oriented Programming (OOP).

Object oriented programming revolves around the concept of objects and class, and four key principle is encapsulation, inheritance, polymorphism, and abstraction, these principles enable code reusability, maintainability, and scalability.

1. What are classes and objects in C++? Provide an example.

In c++, classes and object are fundamental concept of (OOP).

They help in organising code efficiently and making it more modular and reusable.

* Class:- Class is a blueprint

1. What is inheritance in C++? Explain with an example.

Inheritance in c++ is a mechanism that allows a class (derived class) to inherit properties and method to another class (base class)

1. What is encapsulation in C++? How is it achieved in classes?

In c++, encapsulation is achieved by bundling of data and methods into single unit called a Class. This bundling allows data to be protected from direct access and manipulation from outside the class, promoting data hiding and modularity.

1. **Declare Data Members as Private:**

Data members (variables) that are intended to be encapsulated within the class are declared as private. This makes them accessible only to other members of the same class, not to code outside the class.

1. **2. Provide Public Member Functions:**

To allow controlled access and modification of the private data, public member functions are defined within the class. These functions provide a controlled interface for external code to interact with the encapsulated data.

1. **3. Access Specifiers:**

The private and public keywords are access specifiers that control the visibility and accessibility of class members, a key aspect of encapsulation.