PROGRAMMING WITH C & C++
LECTURER NOTES

UNIT1

Introduction to programming.

COBAL, FORTRAN.....etc.

History of 'C' language

ALGOL

CPL

BCPL

B

C-Dennis Ritchie.

Structure of c program

Declaration section

Pre-processor directives

Global declaration section

Main section

Tokens

Variables

Constants

Key words

Identifiers

Operators.

Data types

Primitive data types/Fundamental data types.

Derived data types

User defined data types.

Operators

Unary operators

Binary operators

Ternery operators

Special operators

Standard Input & Output Operations In 'C' Language

Scanf ()

Printf ()

Conditional Statements.

If statements

If-else statements

If-else-if statements

Nested-if statements

Switch statements.

UNIT2

Looping Statements

For loop

While loop

Do while loop

Un-Conditional Statements

Goto statements

Break statements

Continue statements

Arrays

Introduction to arrays

Advantages of arrays

Numeric array

1-DNA, 2-DNA

Character array

1-DCA, 2-DCA

Declaration of array

Initialization of array

Accessing the elements of an array

Assigning the new elements of an array

Manipulating array elements

Multi-dimensional arrays.

UNIT3

Strings

Introduction to strings

Declaration of string variables

Initialization of string variables

String Handling Functions

Strlen ()

Strrev ()

Strcat ()

Strlwr ()

Strupr()

Strcpy ()

Strcmp ()

Functions

Introduction to functions

Advantages of functions

Declaration of functions

Proto type of functions

Syntax of functions

Pre-defined functions

User defined function

Function declaration state

Function calling state

Call-by –value

Call-by-reference

Recursion.

UNIT4

Introduction to object oriented programming

Advantages of object oriented programming

Basic features of oops.

Oops Principles.

Class, Object, Encapsulation, Data Hiding, Data Abstraction, Polymorphism, Inheritance, Message Passing, Dynamic Binding.

Structure of C++ program

Cin & Cout objects

Friend function & friend class

Static functions- instance functions

Instance data member- static data members

Constructors

Initialization Of Objects

Types Of Constructors.

Default/No parameterized constructor

Over loaded constructor

Copy constructor

Dynamic constructor

Destructors

Destruction of object memory.

Operators

Unary operators

UNIT5

Inheritance

Introduction to inheritance

Advantages of inheritance

Base class/Derived class/Defining derived class,

Acess Specifiers

Private

Public

Protected

Types Of Inheritance.

Single Inheritance, Multi-Level Inheritance, Multiple Inheritance, Hierarchical Inheritance, Hybrid Inheritance, Multi Path Inheritance

Programming with C and C++

R Srihari

Introduction of C

- C Is a general-purpose computer programming language. It was created in the 1970s by Dennis Ritchie, and remains very widely used and influential. By design, C's features cleanly reflect the capabilities of the targeted CPUs. It has found lasting use in operating systems, device drivers, protocol stacks, though decreasingly for application software. C is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.
- A successor to the programming language B, C was originally developed at Bell Labs by Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages,with C compilers available for practically all modern computer architectures and operating systems. C has been standardized by ANSI since 1989 (ANSI C) and by the International Organization for Standardization (ISO).

Memory management

- One of the most important functions of a programming language is to provide facilities for managing memory and the objects that are stored in memory. C provides three principal ways to allocate memory for object.
- **Static memory allocation**: space for the object is provided in the binary at compile-time; these objects have an extent (or lifetime) as long as the binary which contains them is loaded into memory.
- **Automatic memory allocation**: temporary objects can be stored on the stack, and this space is automatically freed and reusable after the block in which they are declared is exited.
- **Dynamic memory allocation**: blocks of memory of arbitrary size can be requested at run-time using library functions such as malloc from a region of memory called the heap; these blocks persist until subsequently freed for reuse by calling the library function realloc or free

Arrays

- Array types in C are traditionally of a fixed, static size specified at compile time. The more recent C99 standard also allows a form of variable-length arrays. However, it is also possible to allocate a block of memory (of arbitrary size) at run-time, using the standard library's malloc function, and treat it as an array.
- Since arrays are always accessed (in effect) via pointers, array accesses are typically not checked against the underlying array size, although some compilers may provide bounds checking as an option. Array bounds violations are therefore possible and can lead to various repercussions, including illegal memory accesses, corruption of data, buffer overruns, and run-time.
- C does not have a special provision for declaring multi-dimensional arrays, but rather relies on recursion within the type system to declare arrays of arrays, which effectively accomplishes the same thing. The index values of the resulting "multi-dimensional array" can be thought of as increasing in row-major order.

Introduction of C++

- C++ (pronounced “C plus plus”) is a high-level general-purpose programming language created by Danish computer scientist Bjarne Stroustrup as an extension of the C programming language, or “C with Classes”. The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language, and many vendors provide C++ compilers, including the Free Software Foundation, LLVM, Microsoft, Intel, Embarcadero, Oracle, and IBM, so it is available on many platforms.
- C++ was designed with systems programming and embedded, resource-constrained software and large systems in mind, with performance, efficiency, and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, video games, servers (e.g. E-commerce, web search, or databases), and performance-critical applications (e.g. Telephone switches or space probes).

Inheritance in C++

- The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.
- Inheritance is a feature or a process in which, new classes are created from the existing classes. The new class created is called “derived class” or “child class” and the existing class is known as the “base class” or “parent class”. The derived class now is said to be inherited from the base class.
- **Sub Class:** The class that inherits properties from another class is called Subclass or Derived Class.
- **Super Class:** The class whose properties are inherited by a subclass is called Base Class or Superclass.

Types of Inheritance in C++

- **1. Single Inheritance**: In single inheritance, a class is allowed to inherit from only one class. i.e. One subclass is inherited by one base class only.
- **2. Multiple Inheritance**: Multiple Inheritance is a feature of C++ where a class can inherit from more than one class. i.e one subclass is inherited from more than one base class.
- **3. Multilevel Inheritance**: In this type of inheritance, a derived class is created from another derived class.

Types of inheritance

4. **Hierarchical Inheritance**: In this type of inheritance, more than one subclass is inherited from a single base class. i.e. More than one derived class is created from a single base class.

5. **Hybrid (Virtual) Inheritance**: Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.