UCSH-402

Your progress

Announcements: Assignments/Labs/Class Work

General news and announcements:

Syllabus

Reference Book-I(Click Here)

Prerequisite: C Language

Syllabus Chapters: Chapter 1-2 and Chapter10-19

Reference Book-II

Tasks and Progress Tracker(Keep the Tracker always updated.Thank You...)

Brief History of C++

1. Bjarne Stroustrup developed C++ at AT&T Bell Laboratories as an extension of C in 1980.
2. C was invented by Dennis Ritchie at the same place in the early 1970s.
3. C++ was first installed outside the designer's research group in July 1983.

4. Initially, several C++ features had not been invented.
5. Suggested advantage: Existing C users can gradually upgrade to C++ by feeding their C code through the C++ translator.
6. Disadvantage: Some believe an abrupt change to object-oriented programming is necessary for a proper paradigm shift.
7. C++ evolved from a dialect of C known as "C with Classes."
8. It borrowed key ideas from Simula67 and ALGOL68 programming languages.
9. "C with Classes" lacked operator overloading, references, and virtual functions.
10. The name "C++" was coined by Rick Mascitti in the summer of 1983, indicating evolutionary changes from C.
11. The name "C+" is a syntax error and has been used for an unrelated language.
12. The language is called C++ instead of D to emphasize it as an extension of C.
13. C++ offers improved compile-time type checking and support for modular and object-oriented programming.
14. Prominent C++ features include classes, operator and function overloading, free store management, constant types, references, inline functions, inheritance, virtual functions, streams, templates, and exception handling.

Week-1 & 2- Revision of C and Data Abstraction,Class,Object,cin ,cout,and OOPS paradigms

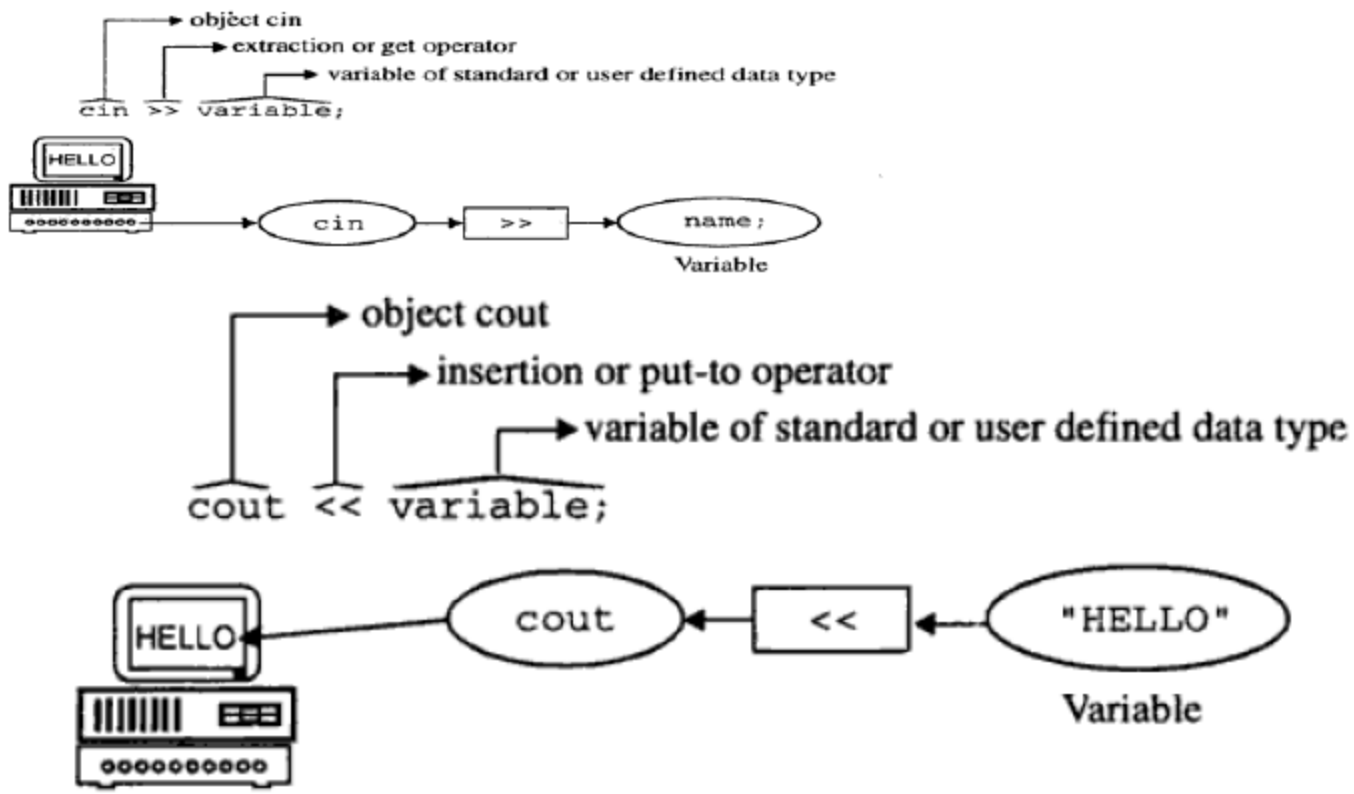
- Pre-Requisite Evaluation: Variables, Operators, Functions, and Arrays.
- i) Doubt on Conditional Statement:Q3-Answer
- Introduction to Object-Oriented Programming: Intuition behind Data Abstraction
- Input and Output in C++ -Programs from the Test-1 implemented in C++
- Structure vs Class

Pre-requisite-Test-1

Pre-Requisite Test-1 Results

Introduction to OOPS-Real World- Data Abstraction-with Examples

Working with cin and cout



First Program Using cin and cout-Detailed Explanation

Class Name:	Test
Member Variables:	Int a
Member Functions:	Void Set(),Void Display()

Object Oriented Paradigms-Read-(Page-21)

Basic definitions:

- Encapsulation
- Inheritance
- Polymorphism

Quiz-2

Basic Questions on Object Oriented Paradigms from the video !!!

Week-3-Designing of complex systems(Page-19)

Here we will have a forum to discuss on
1. Structure and design of Complex systems
2. OOPS and its scope beyond Functional Programming/ Data Decomposition etc.

Forum-1

What is software Complexity?

Week-4-Moving from C to C++

Topics covered:

- Passing Parameters to functions
- Defining functions outside the class
- Inline Functions: (USED ONLY FOR FUNCTIONS DEFINED OUTSIDE THE CLASS)
  1. **FUNCTIONS DEFINED INSIDE THE CLASS BODY ARE BY DEFAULT INLINE IN NATURE**.
  2. Function execution involves the overhead of jumping to and from the calling statement.
  3. The execution time overhead is large for small functions.
  4. Inline functions can be used to reduce this overhead.
  5. A function in C++ can be treated as a macro if the `inline` keyword precedes its definition.
  6. Inline functions substitute the function body at the point of the call instead of making an explicit function call.
  7. This substitution reduces the runtime overhead of the function linkage mechanism.
  8. The function linkage mechanism involves saving actual parameters and the return address onto the stack and cleaning the stack on return.
  9. The linkage process is more costly than including the computation instructions directly in the program.

10. Inline functions combine the flexibility and benefits of modular programming with the speedup of macros.
11. Small inline functions do not significantly increase code size despite being substituted at the call point.
12. It is advisable to define functions with small bodies as inline functions to optimize performance.

- Define a variable at the point of use
- Scope Resolution Operator (::)
- Literals-in C++
- // Integer literals of different bases
    1. int dec= 42;
    2. int oct= 052;
    3. int hex= 0x2A;
    4. int bin= 0b101010;
    5. long int lint= 42L;
    6. unsigned int uint= 42U;
    7. long long int llint= 42LL;
- // float literal
    1. float f = 3.14f;
    2. double d = 3.14;
    3. long double ld = 3.14L
    4. long double sld = 1.22e11;
- //Boolean Literals
    1. bool isTrue = true;
    2. bool isFalse = false;
- Reference Variables (Type and run the two codes in Page 58 and 59)
    1. C++ supports one more type of variable called reference variable, in addition to the value variable and pointer variables of C.
    2. Value variables are used to hold some numeric values;
    3. pointer variables are used to hold the address of (pointer to) some other value variables.
    4. Reference variable behaves similar to both, a value variable and a pointer variable.
    5. A reference variable acts as an alias (alternative name) for the other value variables.
    6. It does not provide the flexibility supported by the pointer variable. Unlike pointer variable, when a reference is bound to a variable, then its binding cannot be changed.
    7. All the accesses made to the reference variable are same as the access to the variable, to which it is bound.
    8. The reference variable must be initialized to some variable only at the point of its declaration.

9. ***Syntax: int & a = b;***
10. Initialization of reference variable after its declaration causes compilation error.

Week-5-Memory Management New and delete operators

C++ provides three kinds of memory allocation for objects:

- Automatic (allocated on the stack)
- Local variables within functions normally use automatic storage.
- Dynamic (allocated from a heap (***new*** operator allocates memory and ***delete*** operator to deallocate memory).
- Dynamic storage is allocated from a heap on an explicit request from the programmer and it must be explicitly released since, standard implementations of C++ do not have a garbage collector.
- Memory Leaks and Dangling Pointers(Avoidance using NULLPTR).
- Example-1
- Example-2
- Example-3 (Will be discussed after undertanding Constructors)
- PROGRAM: Write a program using classes and objects for the follwing task:
- **A Company X has three subbranches (A,B,C) situated at three differnt places.The company possess an initial captial of Rs 50 Lakhs.For the financial year 2025-26,A requires 15 Lakhs,B requires 14 lakhs and C requires 9 Lakhs for their respectie operations.Accordlingly the company allocates the funds required at each sub-branch.After all the allocations ,how much is left with the Company.**
- NOTE:AFTER CODING UPDATE THE PROGRESS SHEET WITH YOUR CODE.
- Static (per-allocated by the compiler in fixed global memory)
- Static storage is obtained by defining a variable outside any function using the static keyword.
- ***All the objects of the class can access the static variable.***

Object as Parameters (Most of the students used in Complex Class)

Static Memory Allocation Programs

Lab Test

Lab Test C++

Submit the code in the progress tracker.

Static variables ,Static Functions,this pointer,Constructors,Destructors

- Global Variable vs Static Variable
- Static Variables are class variables can be accessed from the main function as ***classname::variablename***

Understand and Comment

```
#include <iostream>
using namespace std;
class X {
public:
   int i;
   string Name;
  X(){
     i=100;
     Name="Sai";
  }
};

int main() {
  X x1;
  cout<<x1.i<<endl;
  cout<<x1.Name<<endl;
  return 0;
}
```

Week-8-Theory and Lab

Not available

Week-9-Theory and Lab

Not available

Week-10-Theory and Lab

Not available

Week-11-Theory and Lab

Not available

Week-12-Theory and Lab

Not available

Week-13-Theory and Lab

Not available

Week-14-Theory and Lab

Not available

eGuru

Quick Links

Thought for the day

About Us

Student Support

Contact

Follow Us

Facebook

Contact

Sri Sathya Sai Institute of Higher Learning, Vidyagiri, Prasanthi Nilayam - 515 134, Sri Sathya Sai District, Andhra Pradesh, India.

Data retention summary

Get the mobile app