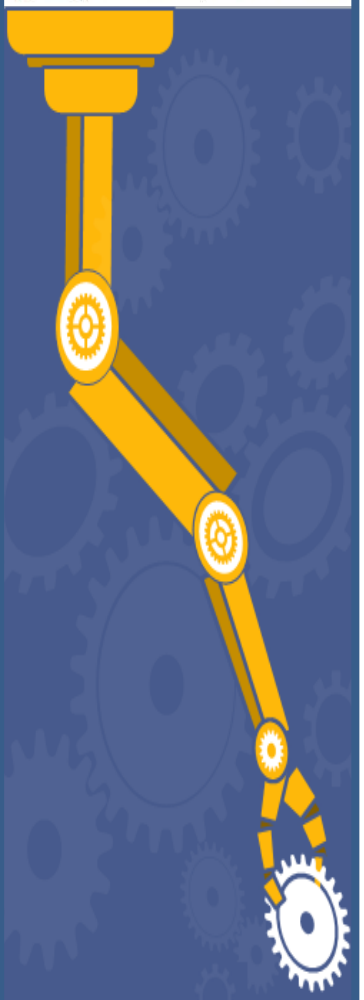
VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)



BCSE102L- Structured and object-oriented programming

Dr. P.Keerthika
Associate Professor
School of Computer Science & Engineering
VIT, Vellore.

BCSE102L- Structured and object-oriented programming

Module:1	**C Programming Fundamentals**	**2 hours**
Variables - Reserved words – Data Types – Operators – Operator Precedence - Expressions - Type Conversions - I/O statements - Branching and Looping: if, if-else, nested if, if-else ladder, switch statement, goto statement - Loops: for, while and do…while – break and continue statements.		
Module:2	**Arrays and Functions**	**4 hours**
Arrays: One Dimensional array - Two-Dimensional Array – Strings and its operations. User Defined Functions: Declaration – Definition – call by value and call by reference - Types of Functions - Recursive functions - Storage Classes - Scope, Visibility and Lifetime of Variables.		
Module:3	**Pointers**	**4 hours**
Declaration and Access of Pointer Variables, Pointer arithmetic – Dynamic memory allocation – Pointers and arrays - Pointers and functions.		
Module:4	**Structure and Union**	**2 hours**
Declaration, Initialization, Access of Structure Variables - Arrays of Structure - Arrays within Structure - Structure within Structures - Structures and Functions – Pointers to Structure -		
Module:5	**Overview of Object-Oriented Programming**	**5 hours**
Features of OOP - Classes and Objects - "this" pointer - Constructors and Destructors - Static Data Members, Static Member Functions and Objects - Inline Functions – Call by reference - Functions with default Arguments - Functions with Objects as Arguments - Friend Functions and Friend Classes.		
Module:6	**Inheritance**	**5 hours**
Inheritance - Types of Inheritance: Single inheritance, Multiple Inheritance, Multi-level Inheritance, Hierarchical Inheritance - Multipath Inheritance - Inheritance and constructors.		
Module:7	**Polymorphism**	**4 hours**
Function Overloading - Operator Overloading – Dynamic Polymorphism - Virtual Functions - Pure virtual Functions - Abstract Classes.		
Module:8	**Generic Programming**	**4 hours**
Function templates and class templates, Standard Template Library.		
	Total Lecture hours:	**30 hours**

BCSE102L- Structured and object-oriented programming – Text Books and Reference Books

Text Book(s)	
1.	Herbert Schildt, C: The Complete Reference, 4th Edition, McGraw Hill Education, 2017
2.	Herbert Schildt, C++: The Complete Reference, 4th Edition, McGraw Hill Education, 2017.
Reference Books	
1.	Yashavant Kanetkar, Let Us C: 17th Edition, BPB Publicaitons, 2020.
2.	Stanley Lippman and Josee Lajoie, C++ Primer, 5th Edition, Addison-Wesley publishers, 2012.

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BCSE102P- Structured and object-oriented programming Laboratory

	Indicative Experiments
1.	Programs using basic control structures, branching and looping
2.	Experiment the use of 1-D, 2-D arrays and strings and Functions
3.	Demonstrate the application of pointers
4.	Experiment structures and unions
5.	Programs on basic Object-Oriented Programming constructs.
6.	Demonstrate various categories of inheritance
7.	Program to apply kinds of polymorphism.
8.	Develop generic templates and Standard Template Libraries.

Text Book(s)	
1.	Robert C. Seacord, Effective C: An Introduction to Professional C Programming, 1st Edition, No Starch Press, 2020.
Reference Book(s)	
1.	Vardan Grigoryan and Shunguang Wu, Expert C++: Become a proficient programmer by learning coding best practices with C++17 and C++20's latest features, 1st Edition, Packt Publishing Limited, 2020.

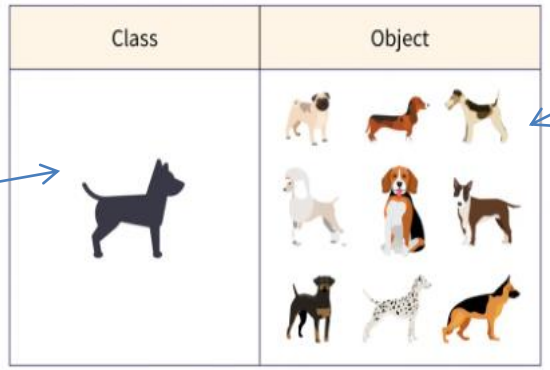
BCSE102L- Structured and Object-Oriented Programming

- **Module-5: Overview of Object Oriented Programming**
  - **Features of OOP- Classes and Objects, This Pointer**
  - **Constructors and Destructors**
  - **Static Data Members and Member Functions**
  - **Inline Functions**
  - **Functions with Default Arguments**
  - **Functions with Objects as Arguments**
  - **Friend Functions and classes**

Class

- A class in C++ is a **user-defined type or data structure** declared with a keyword class that has **data and functions** as its members.
- A class can be used by declaring an object.
- Classes act as a user-defined data type to create objects with similar properties.
- A Class is merely a blueprint of data.
- When a class is created no memory is allocated but when an instance is created by declaring an object, memory is then allocated to store data and perform required functions on them.

Common



Real
Physical

Defining Class in C++

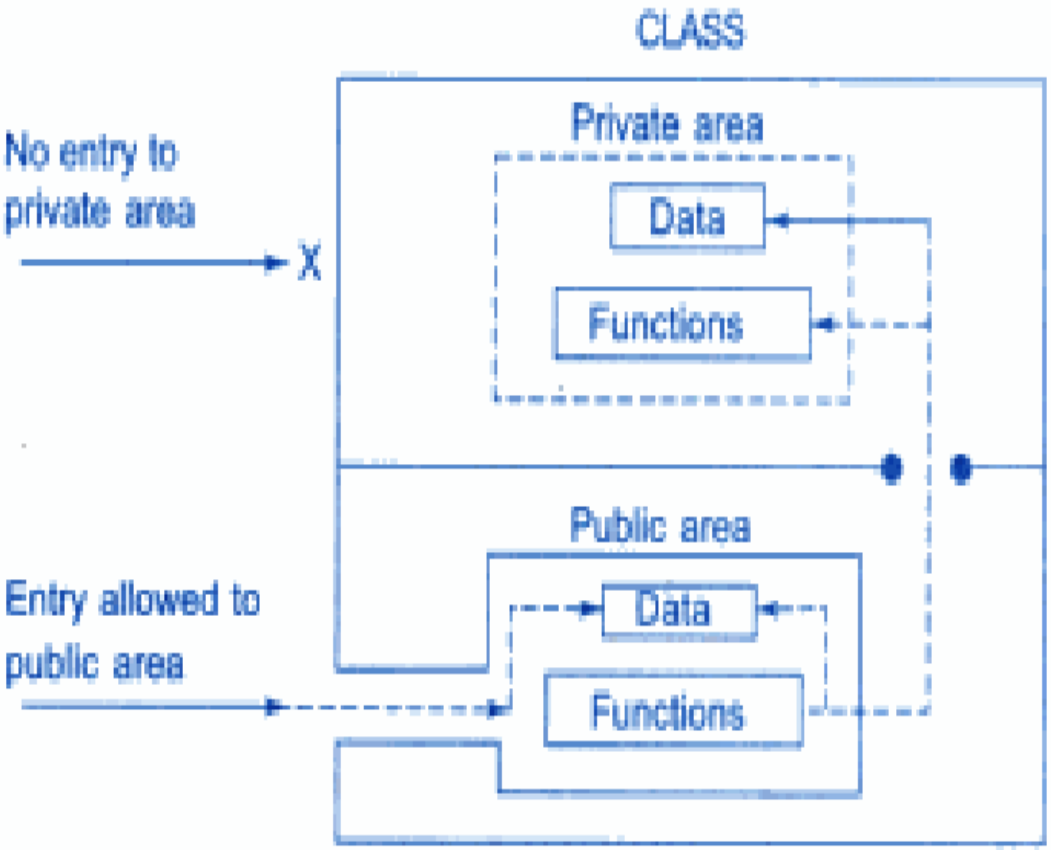
- A Class is defined by a keyword class followed by a class name (user's choice) and a block of curly brackets with semicolons after the block.T
- The block starts with access specifiers followed by data members and member functions.
- Access Specifiers - defines how the members of the class can be accessed.
- **Public:** members can be accessed outside the class.
- **Private:** members cannot be accessed outside the class.
- **Protected:** members cannot be accessed(viewed) from outside the class, but can be accessed in inherited classes(subclasses).

General Form of Class

- A Class is a way to bind the data and its associated functions together.

```
class class name
{
        private:
                variable declaration;
                function declaration;
        public:
                variable declaration;
                function declaration;
};
```

How data is hided in class?



Class - Example

```
class product
{
        int number;
        float cost;
        public:
        void getdata(int a, float b);
        void putdata();
};
```

- *Contains two data members and two function members.*
- *Data members are private by default*
- *Function members are public by declaration*
- *Getdata() – used to assign values to member variables*
- *Putdata()- displaying the values*
- *Note: Data members cannot be accessed by any function that is not a member of class* ***" product"***
- ***Data members are usually declared as private***
- ***Member functions are declared public.***

Creating Objects

- Note: Declaration of " product" – does not define any objects but only specifies what they contain.
- Once class created/declared- we can create variables of that type by using the class name

```
class product
{
        int number;
        float cost;
        public:
        void getdata(int a, float b);
        void putdata();
} x,y, z;
```

Creating objects x,y,z of type product

How to access class members?

- Remember // Private data of a class can be accessed only through the member functions of that class.
- How to access from main function??

```
Object_name.function_name(actual arguments);
```

- Example:

x.getdata(50,12.5); // It assigns 50 to number and 12.5 to cost of the **object x** by implementing getdata() function.

x.putdata() // displays the value

Note: member function can be invoked only by using an object of class(i.e same class).

Defining member functions

- Member functions of a class are used to access, use or modify the data members of that class. We can define member functions in two ways:
  - Outside the class definition
  - Inside the class definition// Inline functions
- Inside: We can define a member function inside the class directly without declaring it first in class.

```
class product
{
        int number;
        float cost;
        public:
        void getdata(int a, float b);
        void putdata()
        {
                cout<<"number";
                cout<<"cost";
        }
};
```

```
class Dog{                          // class ClassName
   public:                          //Access specifiers
      string breed, color;          //Data members

      void displayColor(){          //Member functions
         cout<<color<<" ";
      }

      void displayBreed(){
         cout<<breed<<" ";
      }
};                                  // end with semicolon
```

Defining member functions

- Outside the class definition
- To Define a member function outside the class we need to declare it first inside the class and then define it outside the class according to the following syntax:

```
ReturnType classname::memberfunction(arguements)
{
        Function body
}
```

- Difference between normal function and member function??
- :: ->Scope resolution operator
- Scope of the function is restricted to the class name specified in the header.

Implementation of member functions

```
# include<iostream.h>
Using namspace std;
class product
{
        int number;
        float cost;
        public:
        void getdata(int a, float b);// Function declared
        void putdata()// Function defined in the class itself
        {
                cout<<"number:"<<"number"<<endl;
                cout<<"cost"<<"cost"<<endl;
        }
};
Void product :: getdata(int a, float b) // Membership label
{
        number=a;
        cost=b;
}
```

Implementation of member functions

```
int main()
{
        product x; // object creation
        x.getdata(50,12.5);       // Calling member function
        x.putdata();              // Calling member function

        product y;
        y.getdata(150,100.5);     // Calling member function
        y.putdata();              // Calling member function

        return  0;
}
```

Making outside function inline

```
class product
{
        int number;
        float cost;
        public:
        void getdata(int a, float b);    // Declaration
};

inline void product :: getdata(int a, float b) // Definition
{
        number=a;
        cost=b;
}
```

Nesting of member functions

```
# include<iostream.h>
Using namspace std;
class s
{
        int m,n;
        public:
        void input(void);
        void display(void);
        int largest(void);
};
int s:: largest(void)
{
        if(m>n)
                return m;
        else
                return n;
}
```

```
void s:: input(void)
{
        cin>>m>>n;
}

void s::display(void)
{
   cout<<"Largest value="<<
        largest()<< "\n";
}

int main()
{
        s A;
        A.input();
        A.display();
        return 0;
}
```

Private Member function

```
class sample
{
int number;
void read(void);
public:
void update(void);
void write(void);
};
```

```
Void sample :: update(void)
{
        read(); // no object is used
}
```

- Private member function can only be called by another function that is a member of its class
- Even an object cannot invoke a private function
- S1.read() // *Illegal- objects cannot access private members*
- *read() can be called by the function update() to update the value of number*

Arrays within a class

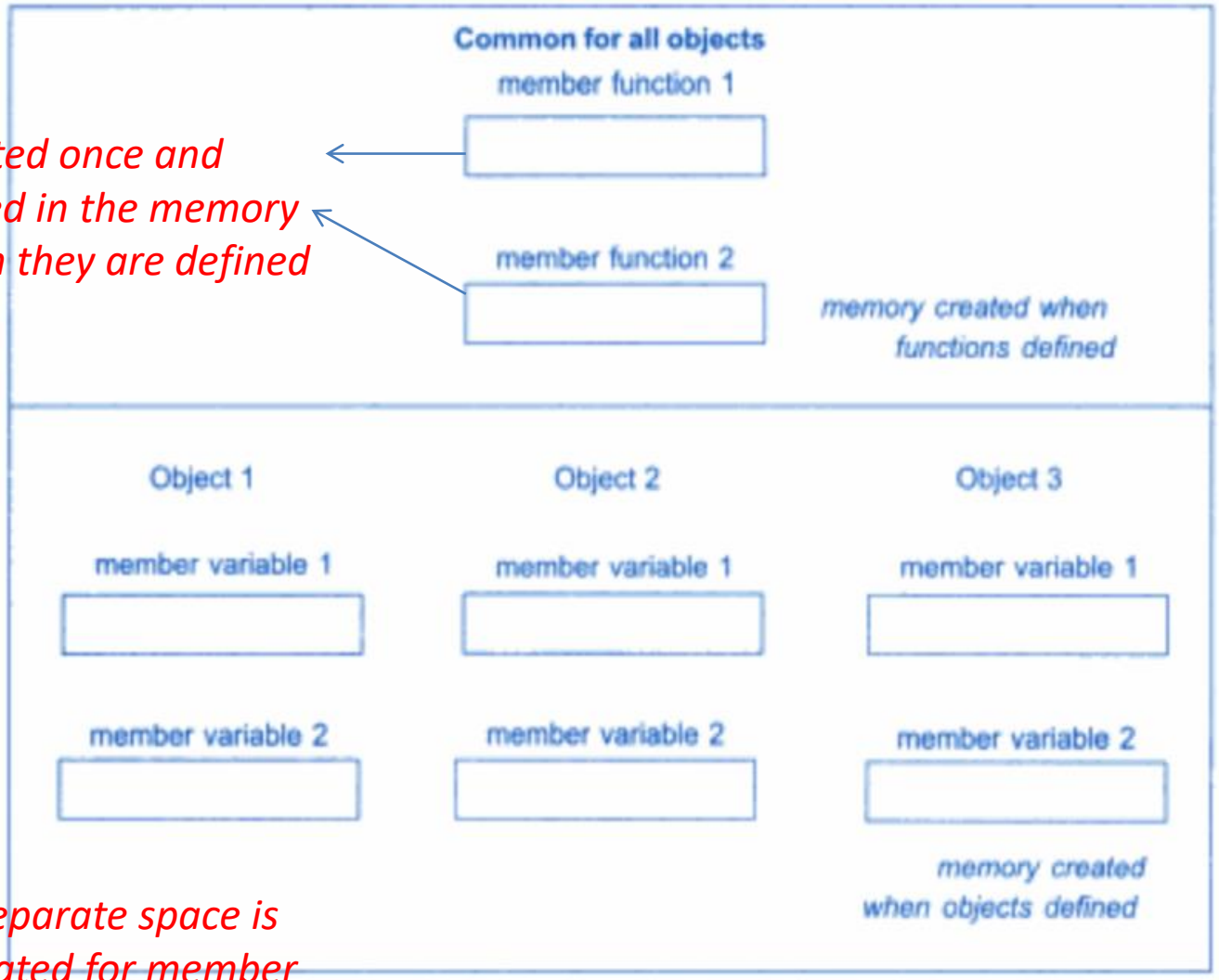
- Arrays can be used as member variables in a class.

```
class product
{
        int number[10]; // type of integer array
        public:
        void getdata(void);
        void putdata();
} ;
```

Example: Processing shopping list
Source: OOP with C++,
E.Balagurusamy

How memory is allocated for objects??

Created once and placed in the memory when they are defined



No separate space is allocated for member functions when objects are created.

This pointer

- Every object in C++ has access to its own address through an important pointer called **this** pointer.
- **this** pointer is an implicit parameter to all member functions.
- Therefore, inside a member function, this may be used to refer to the invoking object.
- Only member functions have **this** pointer.
- **this** pointer is used to access the current object addresses
- **this** pointer is used to distinguish the data members from the local variable when both are declared with the same name.
- So, to identify the data members, we use **this** pointer.

Example: To find the object address

```
#include<iostream>
using namespace std;
Class test
{
int a,b;
Public:
Void show()
{
        a=10;
        b=20;
        Cout<< " obj address = "<< this;  // give current obj address
        Cout<< " a = "<< this->a<<endl;
        Cout<< " b = "<< this->b<<endl;
}
};
Void main()
{
Test t; // created an object
t.show();
}
```

Differentiate data members from local variables

```
#include<iostream>
using namespace std;
Class test
{
int a,b;
Public:
Void show(int x, int y)
{
a=x;
b=y;
}
Void display()
{
Cout<<a<<endl<<b;
}
};
Void main()
{
Test t; // created an object
t.show(10,20);  // 10 is passed to x and 20 is passed to y.
t.display();
}
```

```
#include<iostream>
using namespace std;
Class test
{
int a,b;
Public:
Void show(int a, int b) //same name as data member
{
a=a; //this will create confusion
b=b; // local variable will be given preference so local a will be copied in a. and same with b.
}
Void display()
{
Cout<<a<<endl<<b;
}
};
Void main()
{
Test t; // created an object
t.show(10,20);
t.display(); // print garbage value.
}
```


```
Class test
{
int a, b;
Public:
Void show(int a, int b) //same name as data member
{
        this->a=a; // now first a is data member and second a is local
        variable of the function a
        this-> b=b; // can also be written as (*this).b=b
}
Void display()
{
        Cout<<a<<endl<<b;
}
};
Void main()
{
        test t; // created an object
        t.show(10,20);
        t.display(); // now print a=10 ad b=20.
}
```

Practice Exercises

- Write a C++ program to get and display three student details. Define a Class containing student details like name, reg no and marks.
- Write a C++ program to get and display three student details. Define a Class containing student details like name, reg no and marks. The class members should be declared as private. Functions to get and display the details should be declared inside the class.
- Write a C++ program to get and display three student details. Define a Class containing student details like name, reg no and marks. The class members should be declared as private. Functions to get and display the details should be declared outside the class.
- Write a C++ program to get and display n number of student details. Define a Class containing student details like name, reg no and marks. The class members should be declared as private. Functions to get and display the details should be declared inside the class.
- Write a C++ program to calculate the average marks of five students. Name, reg_no and marks should be used as class members. Assume the data members as private. Array of objects should be considered. Get the input from user through appropriate cin statements.