PAGE NO.:

Advantages of object oriented Language.

↝ We can build the Program from standard working modules that communicate with one another, rather than having to start writing the code from scratch which leads to saving of developments.

↝ OOP system can be easily upgraded from small to large systems.

↝ It is posible that multiple instances of object co-exist without any interference.

↝ It is very easy to partition the works in a project based on objects.

↝ It is possible to map the objects in problem domain to those in the program.

↝ The Principle of data hiding helps the programmer to build secure programs which cannot be invaded by the code in other parts of the program.

↝ Message passing techniques is used for communication between object which makes the interface description with external system much simpler.

PAGE NO.:

↝ OOP language allows to break the program into the bit-sized problem that can be solved easily.

2. Explain the OPPS concept in details

↝ There are some basic concept of the OOP.

1. » Class
2. » Objects
3. » Encapsulation
4. » Abstraction
5. » Polymorphism
6. » Inheritance
7. » Dynamic Binding
8. » Message passing

1. » Class:

It is a user-defined data type, which which holds its own data member and number.

A class is a user-define data type that has data member and function.

Data member are the data variable and member function are the function used to manipulate these variable together these data member of the object in a class.

Q. ➤ Objects :

An object is an identifiable entity with some characteristics and behaviour. An object is an instance of class. When a class is defind, no memory is allocated but when it is instantiated memory is allocated.

```
#include<iostream>
using namespace std;
class person {
        char name[20];
        int id;
    public:
        void getdetails() {}
};
int main()
{
    person P1;
    return 0;
}
```

Object take up space in memory and have an associated address like a record in pascal or structure or union.

3. » Encapsulation :

In normal terms, encapsulation is defined as wrapping up data and information under a single unit.

In object oriented Programming encapsulation is defined as building together the data and the function that manipulate them.

4. » Abstraction

Data abstraction is one of the most essential and important features of object oriented programming in abstraction.

Data abstraction refer to providing only essential information about the data to the outside world implementation.

5. » Polymorphism

The word polymorphism means having many forms. in simple words we can define polymorphism as the ability of a message to be displayed in more than one form.

6. Inheritance :

The capability of a class to derive properties and characters from another class is called inheritance is one of the most important feature of object-oriented Programming.

The class that inherits properties from another class is called sub class or derived class.

The class whose properties are inheritance by a sub-class is called base class.

Inheritance support the concept of "reusibility". we want to create a new class and there is already a class.

7. Dynamic Binding

In dynamic binding, the code to be excuted in responce to the function call is decided at runtime.

Because dynamic binding is flexible it avoids the drawbacks of static binding which connects the function call and definition at buid time.

8. Message Passing

Object communicate with one other by sending and receiving information.

A message for an object is request for the execution of a procedure and therefore will invoke function.

3. Define C In and Out function in details.

1. Defining a function:

A function defining in c programming consists of a function header and a function body.

(i) Return type: A function may return value. the return-type is the data type of the value function return.

(ii) Function name: This is the actual name of the function. the function name and the parameter list together constitute the function signature.

(iii) Parameter: A parameter is like a placeholder. when a function is invoked you pass a value to the parameter.

(iv) Function body: The function body contain a collection of statement that define what the function does.

Q. Declerations function:

→ A function declaration tells the compiler about a fimction name and how to call the function

→ A function declaration
return_type function_name(list);

→ defined function max()
int max(int num1, int num2);

→ Parameter names are not important in function declaration only their type is required so the following is also a valid declaration...
int max(int, int);

3.» Calling function

⇝ A called function performs a defined task and when its return statement is executed or when it function-ending closing brace is reached, it return the Program control back to the main Program.

⇝ To call a function, you simply need to pass the required Parameter along with the function name and if the function return value.

⇝ Example:

```
#include<stdio.h>
int max(int num1, int num2);
int main()
{
   int a=100;
   int b=200;
   int ret;
   ret = max(a,b);
printf("max value is : %d\n", ret);
   return 0;
}
int max(int num1, int2)
{
   int result;
if (num1 > num2)
   result = num1;
else
```

return1 = num2;
return result;
}

output: max value is: 200

4. ➤ Arguments function

If a function is to use arguments it must declare variable that accept the value of the argument. These variable are called the formal parameter of the function.

(i) Call by value:

This method copies the actual value of an argument into the formal parameter of the function.

(ii) Call by reference:

This method copies the address of an argument into the normal parament.

Explain output operators and Input operators details.

7. ➤ iostream: iostream stand for the input-output stream. This header file contain definition of objects like cin, cout, cerr.

2.» iomanip: iomanip stand for input-output manipulators. The method declared in these files are used for manipulating stream. This file contains definition of setw, setprecision.

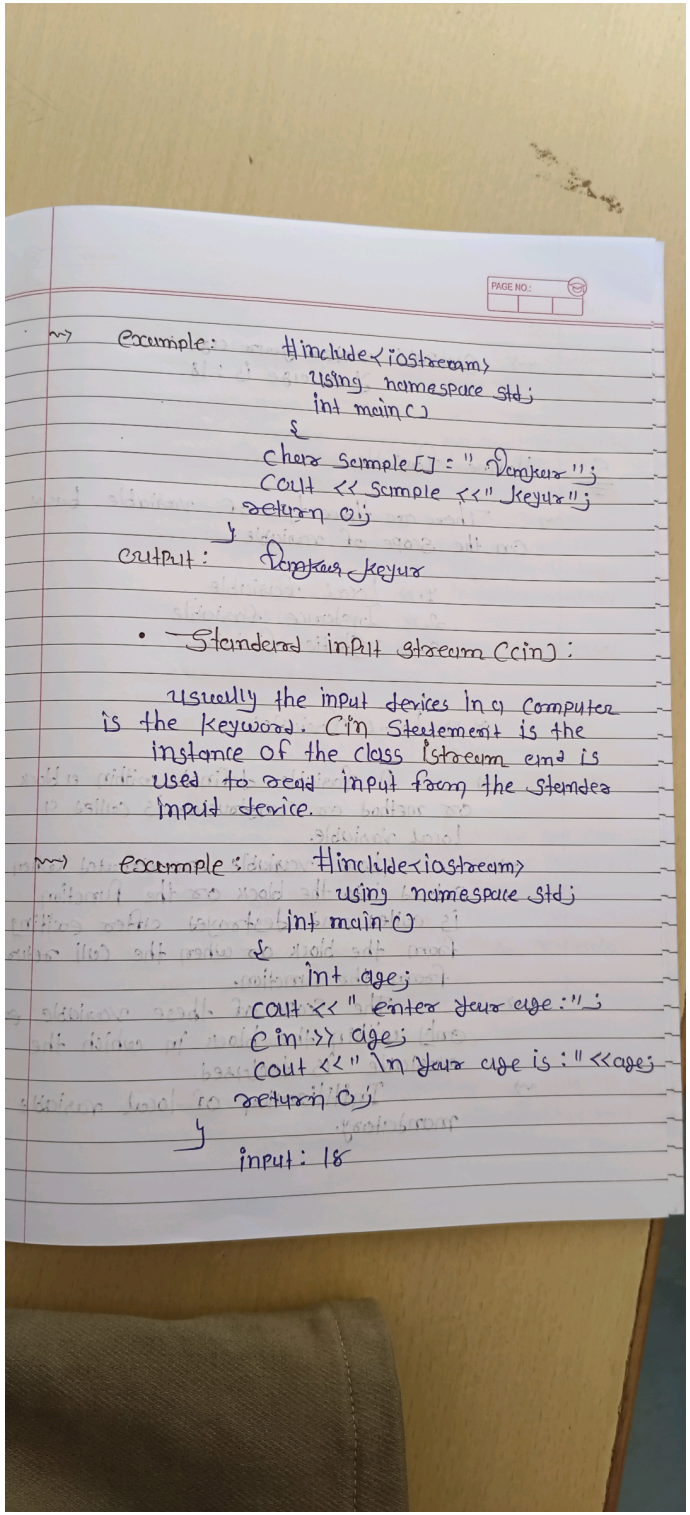
3.» fstream: This header file mainly decribles the file stream. This header file is used to handle the data being read form a file as input or data being written into the file as output.

4.» bits/stdc++: This header file include every standard library. In programming contests, using this file is a good idea, reduce the time waste in header file article.

- Standard output stream (cout):

Usually the standard output device is the display screen. The cout statement is the instance of the ostream class.

It is used to produce output on the standard output device which is usually the display screen.



~> example:

```
#include<iostream>
using namespace std;
int main()
{
char sample[] = "Dongkur";
cout << sample <<" Keyur";
return 0;
}
```

output: Dongkur Keyur

- Standard input stream (cin):

usually the input devices in a computer is the keyword. Cin statement is the instance of the class istream and is used to read input from the standard input device.

~> example:

```
#include<iostream>
using namespace std;
int main()
{
int age;
cout <<" enter your age:";
cin >> age;
cout <<" In your age is :" <<age;
return 0;
}
```

input: 18

~> OUTPUT: Enter your age:
Your age is: 18

Define Variable.

~> There are three types of variable based on the scope of variable.

1. Local Variable
2. Instance Variable
3. Static Variable

1. Local Variable:

~> A variable defined within a block or method or constructor is called a local variable.

~> These variable are created when entred into the block or the function is called and destroyed after exiting from the block or when the call return from the function.

~> The scope of these variable exits only within the block in which the variable is declared.

~> Initilization of local variable is mandatory.

2. Instance Variables:

→ Instance variable are non-static variable and are declared in a class outside any method, constructor, or block.

→ As instance variable are declared in a class, these variable are created when an object of the class is create and destroyed when the object is destroyed.

→ Initialization of instance variable is not mandatory.

→ Instance variable can be accessed only by creating objects.

3. Static Variable:

→ Static variable are also know as class variables.

→ Unlike instance variable we can only have one copy of a static variable per class irrespective of how many object we create.

→ Initilization of static variable is not mandatory.

6. Define Concepts?

→ A concepts is a thought or idea. If you redecorating your bedroom, you might want to start with a concept such as flower garden, or outer space.

7. Explain Scope in details

~> A Scope in any Programming is a region of the program where a defined variable can have its existence and beyond that variable it cannot be accessed.

~> There are three place where variable be declared.

1. Local Variable
2. Global Variable
3. Formal Parameters

1. Local Variable:

~> Variable that are declared inside a function or block are called local variable.

~> They can be used only by statement that are inside that function or block of code.

~> Local variable are not known to function outside their own.

~> Example:

```
#include<stdio.h>
int
#include<iostream>
using namespace std;
int main()
int x = 5;
```

```
cout << "The value of x is :" << x << endl;
return 0;
}
```

Output: The value of x is : 5

2.) Globle Varicible :

↝ A varicible that is declared outside of any function or class and is accessible to all the function and classes in the same file is known as globle varicible.

↝ Example :

```
#include<iostream>
using namespace std;
int globalvar = 10;
void foo()
{
    std :: cout << " The value of
                globlevar inside ";
}
foo();
return 0;
int main()
{
std :: cout << " The value of
                globlevar outside ";
foo();
return 0;
}
```

3.) Formal Parameter.

~> Formal Parameter are treated as local variable within a function and they take precedence over globle variable.

~> Formal parameter on the other hand are variable that are declared in the function definition and are used to hold the actual parameter passed to the function.

~> Example:

```
#include<iostream>
using namespace std;
int multiply(int x, int y)
{
    return x*y;
}
int main()
{
    int a=5;
    int b=7;
    int result = multiply(a,b)
    std::cout<<" The result is:"
    <<result<<std::endl;
    return 0;
}
```

~> Output: The result is: 35