22316

12/10/2023

Page No.
Date

Unit IV : Pointers & Polymorphism
(Weightage - 14 marks)

Questions mandatory

Q1 This pointer
Q2 Explain concept of pointer with suitable example.
Q3 Difference compile time and run time polymorphism.
Q4 WAP to overload unary '-' operator to negate float values of data members and class.
Q5 WAP to overload unary '+' operator to concatenate strings.

Questions

→ Describe function overloading with suitable example. —6 marks

→ List c++ stream classes along with their functions. —2 marks

→ WAP to overload add '+' function to add two integers number & two float numbers —4M

→ Define pointer operator and address operator with example. (4M)

→ WAP to swap two integer numbers using call by reference method. (6M)

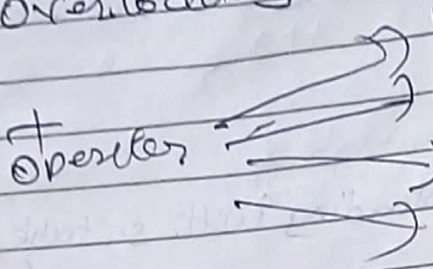
→ With suitable example describe effect of '++' and '--' operators used with pointer in pointer arithematic. (6M)

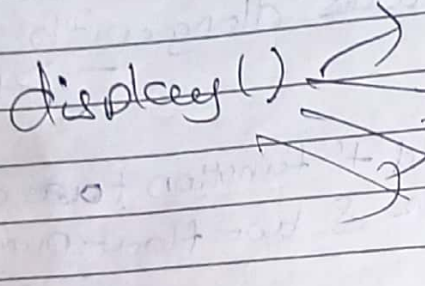
→ WAP c++ program to swap integer no and swap 2 float using function overloading. (4M)

9) Polymorphism

"Poly" stands for many and "morphism" stands for forms and Polymorphism means many forms.

* Polymorphism is the ability of any entity such as a functions or a massage to be processed in more than one form.
* In c++ polymorphism can be achieved either at compiler time or at run time.
* At compiler time polymorphism is implemented using operator overloading & function overloading.





22316

Types of Polymorphism

* ~~Run-tic~~ Compile-time polymorphism.
→ Compile time polymorphism can be implemented by operator overloading and function overloading.
Function overloading: In C++ user can create two or more functions that shared the same name provided their parameter ~~declerations~~ declarations are different. The functions that share the same name are said to be overloaded and the process is known as function overloading

```
main() function
{
    min(6,3);
    .....
    min(min1)
    min(6.6,3.3)
}
```

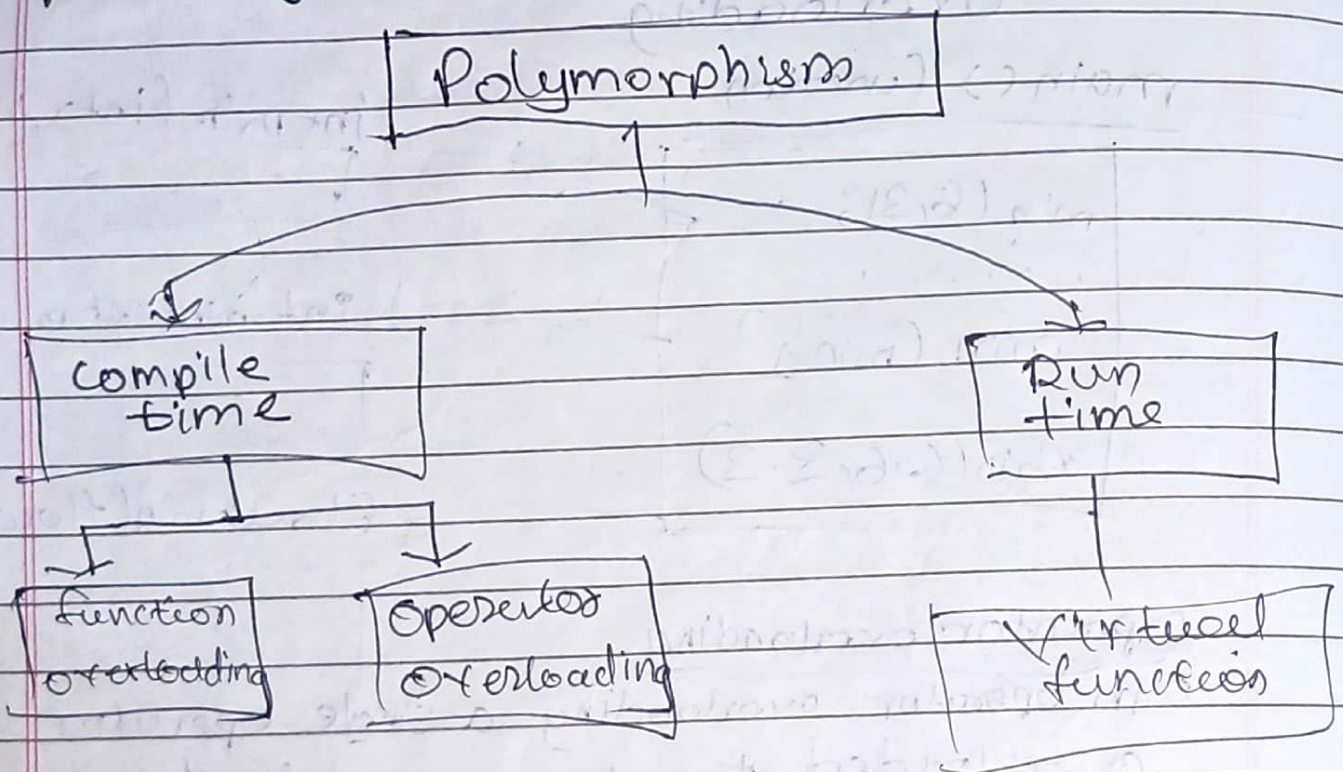
```
int min(int x,int y)
{
}
int min(int m, int n)
{
}
float min(float x, float y)
{
}
```

Operator overloading
: In operator overloading a single operator is overloaded to give user defined meaning to it.

14/9/2023

Run-time polymorphism

* When the required information to call a function is known to the compiler at the execution time or run time is known as run time polymorphism, late bindings as dynamic binding.
* Dynamic Binding is the binding that appropriate function is done at run time.
* The Dynamic Binding is implemented using virtual function.
* Virtual function is the one that does not really exist but appears real to some part of C++ program.



22316

Difference | Polymorphism

Run-time polymorphism	Compile time polymorphism
	Efficient
(1) Calling a function or assigning a value to a variable at run time is called run-time polymorphism.	(1) When compiler acknowledge all the information acquired to called function or all the values of the variables during compile-time it is called as compile-time polymorphism.
(2) In run time polymorphism call is not resolved by the compiler	(2) In compile time polymorphism called is resolved by the compiler.
(3) It is known as Dynamic Binding. late Binding or Run time Binding.	(3) It is also known as static Binding or early Binding or Compile time Binding.
(4) It is achieved by virtual functions.	(4) It is achieved by function overloading and operator overloading.
(5) It provide slow execution as compared to compile time polymorphism.	(5) It provides faster execution because known early at compile time only.
(6) Run-time polymorphism is more flexible as all things execute at run time.	(6) Compile time polymorphism is less flexible as all things execute at compile time.
(7) Example, virtual functions	(7) Examples. overloaded functions Call, Overloaded operators.

10/10/2023

WAP to overload - operator (unary) : to negate the values

```
#include <iostream>
using namespace std;
class Number
{
 int x,y;
 public:
 Number(int a, int b)
 {
  a=x;
  b=y;
 }
 void display()
 {cout<<"value of x = "<<x;
 cout<<"value of y = "<<y;
 void operator-()
  {
   x=-x;
   y=-y;
  }
};
void main()
{
 Number x;
 x.display();
 cout<<"After negation";
 x.display();
}
```

```
cout<<x;
cout<<y;
}
```

22316

```
void main()
{
number x(5,6)
x.display()
x.operator-()
-x
cout<<"After negation"
x.display()
}
```

Q/ WAP to overload binary + operator to concatinate two string.

```
#include<string.h>
#include<iostream>
using namespace std;
class string
{
 char str[20];
 public:
 void getdata()
  {
   cout<<"Enter string";
   cin>>str;
  }
 void operator+()
 {public:
  char str2[10];
  cout<<"Enter string 2";
  cin>>str2;
  cout<<strcat(str,str2);
 }
};
```

```
int main()
{
  String s1, s2;
  s1.getdata();
  s1.operator+();
  s2.getdata();
  s1+s2;
  return 0;
}
```

17/10/23

Describe function overloading with suitable example.

⇒ C++ program to swap two integer numbers and swap two float numbers using function overloading.

```
#include<iostream>
using namespace std;
class swap
{
public:
void swapp(int a, int b)
{int temp;
  temp=a;
  a=b;
  b=temp;
cout<<"Integer after swapping: "<<a<<b
}
void swapp(float x, float y)
  {float tempX;
   tempx= x;
   x= y;
   y= tempx;
```

22316

```
cout<<"float after swapping "<<x<<y
}
};
void main()
{
swap x1;
x1.swapp(3,4);
x1.swapp(3.1,4.1);
}
```

O/P

Integer after swapping 4 3
float after swapping 4.1 3.1

// Write a program overload add function to add two integer number and two float numbers.

```
#include<iostream.h>
class adding
{
int public:
void add(int a,int b)
{
int result;
result=a+b;
cout<<"addition of integer="<<result;
}
void add(int x,int y)
{
float result;
result=x+y;
cout<<"addition of float="<<result
};
```

Overloaded

```
void main()
{
adding x;
x.add(4,5)
x.add(3.1,4.5)
}
```

Output

Addition of integers = 9
Addition of float = 7.6

Virtual functions

Q// Rules for virtual functions - (4 marks)

Virtual functions

* C++ supports, run time polymorphism with the help of virtual functions.
* It is called late or dynamic binding because the appropriate function is selected dynamically at run time.
* A virtual function is member function that is declared which is defined/declared with in base class and redefined by derived class.
* The virtual functions is declared using the ~~key~~ word ~~is~~ virtual.

22316

Page No.
Date

Syntax:

```
class class_name
{
private:
  ---
  ---
public:
virtual return_type function_name1(argument)
virtual return_type function_name2(argument)
};
class derived: public base
{
  return type function_name1()
};
```

Example: virtual void show();

Rules for virtual function.

* The virtual function must be member of same class
* virtual function are accessed by using object pointer.
* They cannot be static member that must be non-static members.
* we can have virtual destructor but not virtual constructor.
* we cannot used pointer to derived class to access object of base type.
* A virtual function can be friend function of other class.
* If virtual function is defined in the base. It is not necessarily defined in derived class.

14/12/2023

* Explain concept of Pointer.

⇒ * A pointer is an powerful technique in C++ programming which helps in accessing data by indirect reference.
* A pointer is an variable which can hold memory address of another variable.
* It is very similar to normal variable and it also has to be defined before using it.
* A pointer is variable whose value is address of another variable.

Advantages

* A pointer improves efficiency of routines in sum complex data structures such as linked lists, stacks, queues, etc..
* With the help of pointer variable can be swap without physically moving them.
* A pointer allows to pass variables, arrays, functions, string, structures as functional arugement (parameters)
* It provides functions which can modify their calling arugement.
* A pointer supports dynamic allocation and deallocation of memory segments.

Syntax : Pointer declaration

* All pointer variable must be declared before it is used in program
* When a pointer variable is declared the variable name must be preceded by asterisk (*)

22316

Syntax

data type * pointer variable;

Example

```
int *ptr;
int *b;
float *a;
char *p;
```

data type - types of pointer variable such (long, int, float etc)

Q11 Define pointer operator and address operator with example.

⇒ Pointer contains memory address which is location of another variable where it has been stored in the memory

A pointer variable consist of two operators namely pointer operator (indirection operator) (*) is a unary operator that returns value of the variable located at the address and address operator (&) this is unary operator.

That returns memory oper address of its operands.

Example Program

```
#include<iostream>
Using namespace std;
int main()
{
  int var;
  int *ptr;
  int val;
  var = 3000;
  ptr = &var;
  val = *ptr;
  cout<<"var"<<var;
  cout<<"ptr"<<ptr;
  cout<<"val"<<val;
}
```

Output

3000 var: 3000
3000 ptr: 0x7dd730
3000 val: 3000

& = address
* = value

22316

23/10/2023

'This' pointer

* We all know that pointer is an variable which holds memory address of another variable.
* Using the pointer technique one can access the data of another variable indirectly.
* The this pointer is a variable which is used to access the address of class itself.
* The member function of every object have access through pointer name this.
* Which points to the object itself.
* This pointer can be treated like any other pointer to an object.

class memory object

Syntax

this → member-identifier/variable;

Example

```
#include <iostream.h>
class Abc
{
    int x;
public:
    void display()
    {
    cout<<"object address="<<this;
    }
};
void main()
{ Abc object, Obj;
Object.display();
obj.display();
}
```

```
void display()
{x=07;
cout<<"x="<<this->x;
}
```

Output
object address:0x557
object address:0x557

The above program creates 2 object (obj ch obj) It display each object's address using this pointer.
The display member function is used to view the display of object address

overloading

① fclose function.

⇒ In c++, a stream refers to seaquence of charachter that are transfer between the program and input-output devices.
The fclose function () closes a stream that are associated with the stream before closing it.
This function deletes all the buffers
if fclose function is not called explicitly the operating system will normally close the file when program execution terminates.

② Explain modes of file operation.

⇒ open() close() read() write()

Open() : Used to create a file.
read() This is used to read the data from file.
write() : Used to write.
close() : Used to close a file.

30/10/2023
Q// Stream classes

22316

200 201 204

21/01/2023

Pointer Codes

Q1 WAP to swap two integers using call by reference.

Q2 With suitable example describe effect of '++' and '--' operators used with pointer in pointer Arithemetic.

Q3 Write c++ program to declare a class train with members as train no & name accept and display data for 1 object of train use pointer to object to call functions of class.

2 With suitable example describe effect of '++' and '--' operator used with pointer in pointer Arithematic.

=> ++ operator: It is reffered as increment operator that increments the value of variable. If ++ operator is used with pointer variable the pointer variable points to next memory address that means Pointer increments with respect to size of data type used to declare pointer variable.

Example

```
int a[5] = {10,20,30,40,50}, *ptr;
ptr = a[0];
for(i=0; i<5; i++)
{
cout<<*ptr;
ptr++;
}
```

In above example ptr pointer points to memory location of a[0]. increment statement '++' increments ptr by memory size of int i.e 2 bytes & now ptr will point to a[1];

ptr declaration print

-- operator

It is refered as decrement operator that decrement values of variable i.e ++ -- decrement variable operator is used with pointer variable the pointer ~~xx~~ points to previous address that means pointer decrement with respect ~~of~~ to size of datatype used to declare pointer variable.

```
int a[5] = {10, 20, 30, 40, 50}; *ptr
ptr = a[4]
for(i=5 4; i>0; i++)
{
    cout<<*ptr;
    ptr--;
}
```

In above example ptr points to memory location of a[4]. decrement statement -- decrements ptr by memory size of int i.e 2 bytes & now ptr will points to a[3];

22316

declaration
memory (address) - print - ✓
Pointer (operator) - swapping
add

Class Call by value

```
#include <iostream>
using namespace std;
void swap(int *p, int *q)
{
    int temp;
    temp = *p;
    *p = *q;
    *q = temp;
}
void main()
{
    int a, b;
    float x, y;
    cout << "enter values of a & b";
    cin >> a >> b;
    cout << "Before swapping ..";
    cout << "A = " << a << "B " << b;
    swap(&a, &b);
    cout << "After swapping ..";
    cout << "A = " << a << "B " << b;
```

Overloading

Bill (20)

Q3 WAP to declare class train with members as train no & name accept and display data for 1 object of train. Use pointer to function class object of class.

⟹

* We can also make a pointer to point an object created by a class. Point to an object created by an class similar to that of normal variables
* A pointer is an variable that holds the address of other variables or object. to
* In pointer to object, the starting address address of object is assigned to pointer.
* After that we can access members of the class with the help of arrow operator

⟹

Rather then . operator

Example

class_name *pointer_name, objectname;
pointer_var = &obj_name;

* In above syntax, the '&' operator returns the starting address of object specified by obj_name;
* The resultant address is assigned to pointer variable specified by the ptr_var

s.getdate()
ptr → get()

```
#include <iostream>
using namespace std;
class train
{
  char tname;
  int train_no;
public:
void get()
{
 cout << "Enter train name & train no :";
 cin >> tname >> train_no;
}
void display()
{
cout << "train name=" << tname;
cout << " train no: " << train_no;
};
void main()
{
train t1, t2, *ptr;
ptr = &t1;
t1.get();
t1.display();
ptr = &t2;
t2.get();
t2.display();
}
```

```
Train* trainptr
trainptr->get();
trainptr->display();
}
```