Lab Code:20ECL203
Programming with C++
Lab Manual

Department of Electronics & Communication Engineering

Bapatla Engineering College :: Bapatla

(Autonomous)

G.B.C. Road, Mahatmajipuram, Bapatla-522102, Guntur (Dist.)
Andhra Pradesh, India.
E-Mail:bec.principal@becbapatla.ac.in
Web:www.becbapatla.ac.in

Contents

Bapatla Engineering College :: Bapatla
(Autonomous)

Vision

- To build centers of excellence, impart high quality education and instill high standards of ethics and professionalism through strategic efforts of our dedicated staff, which allows the college to effectively adapt to the ever changing aspects of education.

- To empower the faculty and students with the knowledge, skills and innovative thinking to facilitate discovery in numerous existing and yet to be discovered fields of engineering, technology and interdisciplinary endeavors.

Mission

- Our Mission is to impart the quality education at par with global standards to the students from all over India and in particular those from the local and rural areas.

- We continuously try to maintain high standards so as to make them technologically competent and ethically strong individuals who shall be able to improve the quality of life and economy of our country.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Vision

To produce globally competitive and socially responsible Electronics and Communication Engineering graduates to cater the ever changing needs of the society.

Mission

- To provide quality education in the domain of Electronics and Communication Engineering with advanced pedagogical methods.

- To provide self learning capabilities to enhance employability and entrepreneurial skills and to inculcate human values and ethics to make learners sensitive towards societal issues.

- To excel in the research and development activities related to Electronics and Communication Engineering.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Educational Objectives (PEO's)

PEO-I: Equip Graduates with a robust foundation in mathematics, science and Engineering Principles, enabling them to excel in research and higher education in Electronics and Communication Engineering and related fields.

PEO-II: Impart analytic and thinking skills in students to develop initiatives and innovative ideas for Start-ups, Industry and societal requirements.

PEO-III: Instill interpersonal skills, teamwork ability, communication skills, leadership, and a sense of social, ethical, and legal duties in order to promote lifelong learning and Professional growth of the students.

Program Outcomes (PO's)

Engineering Graduates will be able to:

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7.Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Specific Outcomes (PSO's)

PSO1: Develop and implement modern Electronic Technologies using analytical methods to meet current as well as future industrial and societal needs.

PSO2: Analyze and develop VLSI, IoT and Embedded Systems for desired specifications to solve real world complex problems.

PSO3: Apply machine learning and deep learning techniques in communication and signal processing.

PROGRAMMING WITH C++ LAB

I B.Tech. II Semester Code:20ECL203/CSL02)

Lectures :	0 Hours/Week	Tutorial :	0 Hours/Week	Practical :	3 Hours/Week
CIE Marks :	30	SEE Marks :	70	Credits :	1.5

Pre-Requisite: None.

Course Objectives: Students will	
➢	Understand advantages of C++ programming over procedural oriented programming learn the basics of variables, operators, control statements, arrays, classes and objects.
➢	Understand, write and implement the following concepts: Inheritance, Interfaces, Packages, Strings and Collections
➢	Understand and write programs on Exception Handling, I/O, and Multithreading
➢	Understand and implement applications using Applets, AWT, Swings and Events

Course Outcomes: At the end of the course, student will be able to	
CO1	Understand basics of variables and operators such as variables, conditional and iterative execution methods etc.
CO2	Identify classes, objects, members of a class and relationships among them needed for a specific problem and Write C++ principles and proper program structuring.
CO3	Demonstrate the concepts of polymorphism, inheritance, packages and interfaces.
CO4	Write C++ to implement error-handling techniques using exception handling

Mapping of Course Outcomes with Program Outcomes & Program Specific Outcomes

	PO's												**PSO's**		
CO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO1	3				3				3						
CO2	3				3				3						
CO3	3				3				3						
CO4		2	2		3				3						
AVG	3	2	2		3				3						

LIST OF EXPERIMENTS

Write C++ programs to illustrate the concept of the following:

1. Arrays
2. Structures
3. Pointers
4. Objects and Classes
5. Console I/O operations
6. Scope resolution and memory management operators
7. Inheritance
8. Polymorphism
9. Virtual Functions
10. Friend Functions
11. Operator overloading
12. Function overloading
13. Constructors and Destructors
14. This pointer
15. File I/O operations

NOTE: A minimum of 10 (Ten) experiments have to be Performed and recorded by the candidate to attain eligibility for Semester End Examination.

Experiment 1: C++ Program on Arrays

Title: Write a C++ Program to display names, roll no's, and grades of 3 Students who have appeared in the examination. Declare the class of name, roll no's and grade. Create an array of class objects. Read and display the contents of the array.

Aim: To write a C program display students grades using arrays.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
#include <iomanip>
using namespace std ;
int main()
{
int k=0;
class stud
{
public:
char name[12];
int rollno;
char grade[2];
};
class stud st[3];
while(k<3)
{
cout<<"Name :" ;
cin>> st[k].name;
cout<<"Roll No. :" ;
cin>>st[k].rollno;
cout<<"Grade: ";
cin>>st[k].grade;
st[k].grade[1]='\0';
puts("press any key…");
k++;
}
k=0;
cout<<"\n Name\t Rollno \t Grade\n";
while(k<3)
{
cout<<st[k].name<<setw(6)<<st[k].rollno<<setw(6)<<st[k].grade<<"\n";
```

```
k++;
}
return(0);
}
```

Expected Output:

```
Name :sai
Roll No. :25
Grade: A
press any key…
Name :KUMAR
Roll No. :36
Grade: A
press any key…
Name :PRAVEEN
Roll No. :46
Grade: A+
press any key…

 Name       Rollno    Grade
sai           25          A
KUMAR      36          A
PRAVEEN    46          A
```

Result:

The C program successfully read the student details through arrays displays the same using arrays.

Experiment 2: C++ Program on Structure

Title: **Program to assign data to members of a structure variable**

Aim: To write a C++ program to define a structure and assign values to members of the structure variable.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;
struct Person
{  string first_name;
   string last_name;
   int age;
   float salary;   };
int main()
{    Person p1;
   cout << "Enter first name: ";
   cin >> p1.first_name;
   cout << "Enter last name: ";
   cin >> p1.last_name;
   cout << "Enter age: ";
   cin >> p1.age;
   cout << "Enter salary: ";
   cin >> p1.salary;
   cout << "\nDisplaying Information." << endl;
   cout << "First Name: " << p1.first_name << endl;
   cout << "Last Name: " << p1.last_name << endl;
   cout << "Age: " << p1.age << endl;
   cout << "Salary: " << p1.salary;
   return 0;
}
```

Expected Output:

```
Enter first name: ravi
Enter last name: kumar
Enter age: 25
Enter salary: 50000
Displaying Information.
First Name: ravi
Last Name: kumar
Age: 25
Salary: 50000
```

Result:
The C++ program successfully read the employee details using structure variables displays the same.

Experiment 3: C++ Program on pointers

Title: **Write a c++ program to access arrays using pointers.**

Aim: To write a C program display values using pointers.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;
int main()
{
   float arr[3]={2.5,3.4,5.6};
   // declare pointer variable
   float *ptr;
   cout << "Displaying the values using pointers: " << endl;
   // use for loop to print values of all array elements
    // ptr = &arr[0]
   ptr = arr;
   for (int i = 0; i < 3; ++i)
   {
      cout << "&arr[" << i << "] = " << *ptr << endl;
ptr++;
   }
   cout<<"\nDisplaying address using pointers: "<< endl;
   // use for loop to print addresses of all array elements
   // using pointer notation
   for (int i = 0; i < 3; ++i)
   {
      cout << "ptr + " << i << " = "<< ptr + i << endl;
   }
   return 0;
}
```

Expected Output:

```
Displaying the values using pointers:
&arr[0] = 2.5
&arr[1] = 3.4
&arr[2] = 5.6

Displaying address using pointers:
ptr + 0 = 0x7ffce45f42a0
ptr + 1 = 0x7ffce45f42a4
ptr + 2 = 0x7ffce45f42a8
```

Result:

The C program successfully acces the values and display the address using pointers.

Experiment 4: C++ Program on Objects and Classes

Title: Given that an EMPLOYEE class contains following members: data members: Employee number, Employee name, Basic, DA, IT, Net Salary and print data members. Write a C++ program to read the data of N employee and compute Net salary of each employee (DA=52% of Basic and Income Tax (IT) =30% of the gross salary) .

Aim: To write a C++ program by creating classes and objects .

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;
class employee
{
char name[10];
int no;
float basic;
float da;
float it;
float ns;
float gs;
public:
void input()
{
cout <<"Enter number:";
cin >> no;
cout <<"Enter name:";
cin >> name;
cout <<"Enter salary:";
cin >> basic;
}
void calculate()
{
da = 0.52 * basic;
gs = da + basic;
it = 0.3 * gs;
ns = gs-it;
}
void output()
{
cout<<no<<'\t'<<name<<'\t'<<basic<<'\t'<<ns<<'\t'<<gs <<'\n';
}
};
```

```
int main()
{
employee emp[20];
int n,i;
cout << "Enter no of employees:";
cin >> n;
for(i=0;i<n;i++)
{
emp[i].input();
emp[i].calculate();
}
cout<<"NUMBER"<<'\t'<<"NAME"<<'\t'<<"BASIC"<<'\t'<<"NET"<<'\t'
<<"GROSS" << "\n";
for(i=0;i<n;i++)
{
emp[i].output();
}
return(0);
}
```

Expected Output:

```
Enter no of employees:2
Enter number:24
Enter name:sai
Enter salary:5000
Enter number:45
Enter name:praveen
Enter salary:5000
NUMBER    NAMEBASIC        NET   GROSS
24     sai    5000  5320  7600
45     praveen      5000  5320  7600
```

Result: The C ++ program successfully acces the values using objects.

Experiment 5: C++ Program on Console I/O operations

Title: **Write a C++ to illustrate the concepts of console I/O operations.**

Aim: To write a C++ program to generate the required output using console I/O operations .

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
cout.width(5);
cout<<"A";
cout.width(15);
cout<<"B";
cout.precision(4);
cout<<3.1452;
cout.fill('/');
cout.width(20);
cout<<"WEL";
cout.fill('-');
cout.width(10);
cout<<"Done";
return(0);
}
```

Expected Output:

```
A          B3.145//////////////////WEL------Done
```

Result: The C ++ program successfully executed using console I/O operations.

Experiment 6a: C++ Program on Scope resolution

Title: **Write a C++ program to use scope resolution operator. Display the various values of the same variables declared at different scope levels.**

Aim: To identify the scope of a variable at different locations in a program in C++ .

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;
int m=10;
int main() {
   int m=20 ;
   { int k=m;
   int m=30;
   cout<< "we are in inner block";
   cout<<"k="<<k<<endl;
   cout<<"m="<<m<<endl;
   cout<<":: m="<<:: m<<endl;
   }
   cout<<"\n we are in outer block \n";
   cout<<"m="<<m<<endl;
   cout<<":: m="<<:: m<<endl;
   return(0);
}
```

Expected Output:

```
we are in inner blockk=20
m=30
:: m=10

 we are in outer block
m=20
:: m=10
```

Result: The C ++ program successfully access variable at different levels in a program.

Experiment 6b: C++ Program on Memory Management Operators

Title: **Write a C++ program to assign memory based on the requirement of programmer.**

Aim: To allocate memory to the variable before assigning values and de allocates after it .

Apparatus Required:

- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
include <iostream>
using namespace std;
int main() {
    int *pointInt; // declare an int pointer
    float *pointFloat; // declare a float pointer
  // dynamically allocate memory
  pointInt = new int;
  pointFloat = new float;
  // assigning value to the memory
  *pointInt = 45;
  *pointFloat = 45.45f;
  cout << *pointInt << endl;
  cout << *pointFloat << endl;
// deallocate the memory
  delete pointInt;
  delete pointFloat;
  cout << "after delallocate the memory" << endl;
  cout << *pointInt << endl;
  cout << *pointFloat << endl;
  return 0;
}
```

Expected Output:

```
45
45.45
after delallocate the memory
0
1.29024e-38
```

Result: The C ++ program successfully allocates and de allocates memory to variables in a program.

Experiment 7: C++ Program on Inheritance

Title: **Write a C++ program to create multilevel inheritance. Create classes A1,A2, A3**.

Aim: To access different variable using inheritance by creating 3 levels .

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include<iostream>
using namespace std;
class A1
{
protected:
char name[15];
int age;
};
class A2 : public A1
{
protected:
float height,weight;
};
class A3: public A2
{
protected :
char sex;
public:
void get()
{
cout<<"Name :";
cin>>name;
cout<<"Age : ";
cin>>age;
cout<<"Sex : ";
cin>>sex;
cout<<"Height : ";
cin>>height;
cout<<"Weight : ";
cin>>weight;
}
void show()
{
cout<<"\nName : "<<name;
```

```
cout<<"\nAge : "<<age<<"Years";
cout<<"\nHeight : "<<height<<"Feets";
cout<<"\nSex : "<<sex;
cout<<"\n Weight : "<<weight<<"kg";
}
};
int main()
{
A3 x;
x.get();
x.show();
return(0);
}
```

Expected Output:

Name :sai
Age : 35
Sex : m
Height : 150
Weight : 87

Name : sai
Age : 35Years
Height : 150Feets
Sex : m
Weight : 87kg

Result: The C ++ program successfully verify the access and assign of variables using inheritance.

Experiment 8: C++ Program on Polymorphism

Title: **Write a C++ program to create an array of pointers. Invoke functions using array objects.**

Aim: To verify the polymorphism using pointers by creating array of objects.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include<iostream>
using namespace std;
class A
{
public:
virtual void show()
{
cout<<"A\n";
}
};
class B:public A
{
public:
void show()
{
cout<<"B\n";
}
};
class C:public A
{
public:
void show()
{
cout<<"C\n";
}
};
class D:public A
{
public:
void show()
{
   cout<<"D\n";
}
};
class E:public A
```

```
{
public:
void show()
{
cout<<"E";
}
};
int main()
{
A a;
B b;
C c;
D d;
E e;
A *pa[] = { &a,&b,&c,&d,&e};
for(int j=0;j<5;j++)
pa[j]->show();
return(0);
}
```

Expected Output:

```
A
B
C
D
E
```

Result: The C ++ program successfully exhibits polymorphism.

Experiment 9: C++ Program on virtual function

Title: **Write a C++ program to demonstrate the working of virtual function.**

Aim: To know the working of virtual function by using base classes.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include<iostream>
using namespace std;
class one
{
public:
virtual void A()
{
cout<<"base  A"<<endl;
}
void B()
{
cout<<"base  B"<<endl;
}
};
class two:public one
{
public:
void A()
{
cout<<"derived A"<<endl;
}
void B()
{
cout<<"derived B"<<endl;
}
};
int main()
{
one obj1;
two obj2;
one *ptr;
ptr=&obj1;
ptr->A();
ptr->B();
ptr=&obj2;
```

```
ptr->A();
ptr->B();
return 0;
}
```

Expected Output:

base A
base B
derived A
base B

Result: The C ++ program successfully exhibits the derived A class and defines the virtual function.

Experiment 10: C++ Program on friend function

Title: **Write a C++ program to demonstrate the working of friend function.**

Aim: To know the working of friend function by using base classes.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;
class Distance {
   private:
      int meter;
      // friend function
      friend int addFive(Distance);

   public:
      Distance() { meter=0; }
};
// friend function definition
int addFive(Distance d) {
   //accessing private members from the friend function
   d.meter += 5;
   return d.meter;
}
int main() {
   Distance D;
   cout << "Distance: " << addFive(D);
   return 0;
}
```

Expected Output:

Distance: 5

Result: The C ++ program successfully exhibits the definition of friend function.

Experiment 11: C++ Program on operator overloading

Title: **Write a C++ program to overload "– "operator .**

Aim: To understood the concept of overloading using "-" operator.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include<iostream>
using namespace std;
class num
{
private:
int a,b,c,d;
public:
num(int x,int y,int z,int w) // constructor
{
a=x;
b=y;
c=z;
d=w;
}
void show(void);
void operator-();
};
void num :: show()
{
cout<<"A= "<<a <<"B= "<<b <<"C="<<c <<"D="<<d;
}
void num :: operator-()
{
a= -a;
b= -b;
c= -c;
d= -d;
}
int main()
{
num X(2,2,8,4); // object define and constructor calling
cout<<"\n Before negation of X: ";
X.show();
-X;
cout<<"\n After negation of X : ";
```

```
X.show();
return 0;
}
```

Expected Output:

Before negation of X: A= 2B= 2C=8D=4
After negation of X : A= -2B= -2C=-8D=-4

Result: The C ++ program successfully exhibits the overloading of '-' operator.

Experiment 12: C++ Program on function overloading

Title: **Write a C++ program to overload a function .**

Aim: To understood the concept of overloading using “add” function.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;

// Function to add two integers
int add(int a, int b) {
    return a + b;
}

// Function to add three integers
int add(int a, int b, int c) {
    return a + b + c;
}

// Function to add two floating-point numbers
float add(float a, float b) {
    return a + b;
}

int main() {
    cout << "Addition of two integers: " << add(5, 3) << endl;
    cout << "Addition of three integers: " << add(1, 2, 3) << endl;
    cout << "Addition of two floats: " << add(2.5f, 3.5f) << endl;

    return 0;
}
```

Expected Output:

```
Addition of two integers: 8
Addition of three integers: 6
Addition of two floats: 6
```

Result: The C ++ program successfully exhibits the overloading add function.

Experiment 13: C++ Program on constructor and destructor

Title: Write a C++ program to demonstrate constructor overloading along with destructor.

Aim: To understood the concept constructor and destructor.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
using namespace std;
class Person {
   private:
    int age;
   public:
    // 1. Constructor with no arguments
    Person() {
        age = 20;
    }
    // 2. Constructor with an argument
    Person(int a) {
        age = a;
    }
    int getAge() {
        return age;
    }
    ~Person()
{
cout<<"\n Destructor invoked";
 cout<<"AGE="<<age;
}
};
int main() {
    Person person1, person2(45);
    cout << "Person1 Age = " << person1.getAge() << endl;
    cout << "Person2 Age = " << person2.getAge() << endl;
    return 0;
}
```

Expected Output:

```
Person1 Age = 20
Person2 Age = 45

 Destructor invokedAGE=45
 Destructor invokedAGE=20
```

Result: The C ++ program successfully exhibits the concepts of constructor and destructor.

Experiment 14: C++ Program on THIS pointer

Title: Write a C++ program using this pointer.

Aim: To understood the concept "this" pointer.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include<iostream>
using namespace std;
class Test
{
private:
int x;
int y;
public:
Test(int x = 0, int y = 0) { this->x = x; this->y = y; }
Test &setX(int a) { x = a; return *this; }
Test &setY(int b) { y = b; return *this; }
void print() { cout << "x = " << x << " y = " << y << endl; }
};
int main()
{
Test obj1(5, 10);
// Chained function calls. All calls modify the same object
// as the same object is returned by reference
obj1.print();
obj1.setX(10).setY(20);
obj1.print();
return 0;
}
```

Expected Output:

x = 5 y = 10

x = 10 y = 20

Result: The C ++ program successfully access the pointer using This pointer.

Experiment 15: C++ Program on file I/O opertions

Title: **Write a C++ program to access files.**

Aim: To understood the concept reading and writing the file.

Apparatus Required:
- Code::Blocks Software
- Computer system with GCC compiler

C Code:

```
#include <iostream>
#include <fstream>  // For file I/O operations
using namespace std;

int main() {
   // 1. Writing to a file
   ofstream outFile("example.txt");  // Create and open a file for writing
   if (!outFile) {
      cerr << "Error opening file for writing!" << endl;
      return 1;
   }
   outFile << "Hello, this is a simple text file." << endl;
   outFile << "File I/O in C++ is easy!" << endl;
   outFile.close();  // Close the file after writing

   cout << "Data written to 'example.txt' successfully!" << endl;

   // 2. Reading from the file
   ifstream inFile("example.txt");  // Open the file for reading
   if (!inFile) {
      cerr << "Error opening file for reading!" << endl;
      return 1;
   }

   string line;
   cout << "\nReading from 'example.txt':" << endl;
   while (getline(inFile, line)) {  // Read line by line
      cout << line << endl;
   }
   inFile.close();  // Close the file after reading

   return 0;
}
```

Expected Output:

```
Data written to 'example.txt' successfully!

Reading from 'example.txt':
Hello, this is a simple text file.
File I/O in C++ is easy!
```

Result: The C ++ program successfully access files.