Object Orientated Programming Lab

Amneesh Singh I6

December 26, 2022

Contents

Programs are followed by their respective outputs (and inputs)

1 Lab 1

1.1 Write a program to take name, address as a character array, age as int , salary as float and contains inline functions to set the values and display it in class named person.

```
#include <iostream>
#include <string.h>

class Person {
private:
  static const std::size_t MAX_NAME = 30;
  static const std::size_t MAX_ADDRESS = 30;
  char name[MAX_NAME];
  char address[MAX_ADDRESS];
  unsigned int age;
  float salary;

public:
  void getName();
  void getAddress();
  void getAge();
  void getSalary();
  void setName(const char[30], size_t);
  void setAddress(const char[100], size_t);
  void setAge(unsigned int);
  void setSalary(float);
};

inline void Person::getName() {
  std::cout << name << std::endl;
}

inline void Person::getAddress() {
  std::cout << address << std::endl;
}

inline void Person::getAge() {
  std::cout << age << std::endl;
}

inline void Person::getSalary() {
  std::cout << salary << std::endl;
}

inline void Person::setName(const char name[], size_t size) {
  strncpy(this->name, name, (size <= MAX_NAME ? size : MAX_NAME - 1));

  if (size > MAX_NAME)
    this->name[MAX_NAME - 1] = '\0';
}
```

```
inline void Person::setAddress(const char address[], size_t size) {
  strncpy(this->address, address, (size <= MAX_ADDRESS ? size : MAX_ADDRESS - 1));

  if (size > MAX_NAME)
    this->name[MAX_NAME - 1] = '\0';
}

inline void Person::setAge(unsigned int age) {
  this->age = age;
}

inline void Person::setSalary(float salary) {
  this->salary = salary;
}

int main() {
  Person person;
  const char name[] = "Retard Singh";
  const char address[] = "420 RetardVille";
  unsigned int age = 44;
  float salary = 4.44;

  person.setName(name, sizeof(name));
  person.setAddress(address, sizeof(address));
  person.setAge(age);
  person.setSalary(salary);

  person.getName();
  person.getAddress();
  person.getAge();
  person.getSalary();

  return 0;
}
```

```
Retard Singh
420 RetardVille
44
4.44
```

1.2 Using the concept of function overloading.Write function for calculating the area of triangle ,circle and rectangle.

```cpp
#include <cmath>
#include <iostream>

// Triangle
double area(double a, double b, double c) {
  double s = (a + b + c) / 2;
  return std::sqrt(s * (s - a) * (s - b) * (s - c));
}

// Rectangle
double area(double a, double b) { return a * b; }

// Circle
double area(double a) { return M_PI * a * a; }

int main() {

  double a = 3, b = 4, c = 5;

  std::cout << "Area of a triangle with sides " << a << ", " << b << " and "
            << c << ": " << area(a, b, c) << std::endl
            << "Area of rectangle with sides " << a << " and " << b << ": "
            << area(a, b) << std::endl
            << "Area of circle with radius " << a << ": " << area(a)
            << std::endl;

  return 0;
}
```

```
Area of a triangle with sides 3, 4 and 5: 6
Area of rectangle with sides 3 and 4: 12
Area of circle with radius 3: 28.2743
```

1.3 Write a program to find number m to power n. The function power takes a double value for m and int value for n. Use default value for n to make the function to calculate squares when this argument is omitted.

```cpp
#include <cmath>
#include <iostream>

double uexponent(double m, unsigned int n) {
  double ret;

  if (!n)
    return 1;

  if (n == 1)
    return m;

  ret = uexponent(m, n / 2);
  ret *= ret;

  if (n % 2)
    return ret * m;
  else
    return ret;
}

double exponent(double m, int n = 2) {
  if (n >= 0)
    return uexponent(m, n);

  return 1 / uexponent(m, std::abs(n));
}

int main() {

  double m = 4.1;
  int n = 4;

  std::cout << m << " raised to " << n << ": " << exponent(m, n) << std::endl
            << "When n is omitted: " << exponent(m) << std::endl;
}
```

```
4.1 raised to 4: 282.576
When n is omitted: 16.81
```

2 Lab 2

2.1 Write a program for multiplication of two matrices using OOP. [Hint: Declare class maths and then define variables and functions inside the class].

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

class Matrix {
private:
  unsigned int r, c;
  vector<vector<int>> matrix;

public:
  Matrix() = default;
  Matrix(int, int);
  void init(vector<vector<int>>);
  unsigned int getRows();
  unsigned int getColumns();
  Matrix operator*(Matrix);
  void display();
};

Matrix::Matrix(int r, int c) {
  this->r = r, this->c = c;
  this->matrix = vector<vector<int>>(r, vector<int>(c));
}

void Matrix::init(vector<vector<int>> src) {
  unsigned int i, j;

  if (this->r > src.size())
    exit(1);

  for (i = 0; i < this->r; i++) {
    if (c > src[i].size())
      exit(1);

    for (int j = 0; j < this->c; j++)
      this->matrix[i][j] = src[i][j];
  }
}

inline unsigned int Matrix::getRows() { return r; }

inline unsigned int Matrix::getColumns() { return c; }

Matrix Matrix::operator*(Matrix operand) {
  unsigned int i, j, k;
  unsigned int r, c;
  unsigned int common;
  Matrix m;
```

```
  if (this->getColumns() != operand.getRows())
    exit(1);

  r = this->getRows();
  c = operand.getColumns();

  m = Matrix(r, c);

  common = this->getColumns();

  for (i = 0; i < r; i++) {
    for (j = 0; j < c; j++) {
      int s = 0;
      for (k = 0; k < common; k++)
        s += this->matrix[i][k] * operand.matrix[k][j];
      m.matrix[i][j] = s;
    }
  }

  return m;
}

void Matrix::display() {
  unsigned int i, j;

  for (i = 0; i < this->r; i++) {
    for (j = 0; j < this->c; j++)
      cout << this->matrix[i][j] << ' ';
    cout << endl;
  }
}

int main() {
  Matrix a(2, 3), b(3, 4);
  Matrix m;

  a.init(vector<vector<int>>{{1, 2, -3}, {3, 2, 1}});

  b.init(
      vector<vector<int>>{{1, -4, -4, 1}, {0, 4, 4, -34}, {3, 1231, 0, -9653}});

  m = a * b;

  m.display();

  return 0;
}
```

```
-8 -3689 4 28892
6 1227 -4 -9718
```

2.2 Create a class TIME with members hours, minutes, seconds. Take input, add two time objects passing objects to function and display the result. [Hint: use class object as parameter of function add()]

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

class Time {
private:
  uint h, m, s;

public:
  Time() = default;
  Time(uint, uint, uint);
  Time operator+(Time);
  void display();
};

Time::Time(uint h, uint m, uint s) {
  if (m > 59 || s > 59)
    exit(1);

  this->h = h;
  this->m = m;
  this->s = s;
}

void Time::display() {
  cout << "Hours: " << this->h << endl
       << "Minutes: " << this->m << endl
       << "Seconds: " << this->s << endl;
}

Time Time::operator+(Time op) {
  Time t;

  t.h = this->h + op.h;
  t.m = this->m + op.m;
  t.s = this->s + op.s;

  t.m += t.s / 60;
  t.s %= 60;

  t.h += t.m / 60;
  t.m %= 60;

  return t;
}

int main() {
  Time t;
  Time a(64, 32, 7);
  Time b(3, 59, 53);
```

```
  t = a + b;

  t.display();

  return 0;
}
```

```
Hours: 68
Minutes: 32
Seconds: 0
```

2.3 **Create a class Student which has data members as name, branch, roll no, age , gender, marks in five subjects. Display the name of the student and his percentage who has more than 70%.Use array of objects size 5.[Hint: also declare percentage variable and calculate it and then compare all the object of student class to find name of student with percentage > 70]**

```
#include <iostream>

using namespace std;

const uint NSUBS = 5;
const uint NSTUDENTS = 5;

class Student {
private:
  char name[30];
  char branch[20];
  uint enrollment;
  uint age;
  char gender;
  uint marks[NSUBS];
  double percentage;

public:
  void input();
  char *getName();
  double getPercentage();
};

void Student::input() {
  uint s = 0, i;

  cout << "Name: ";
  cin >> this->name;
  cout << "Branch: ";
  cin >> this->branch;
  cout << "Enter Enrollment number: ";
  cin >> this->enrollment;
  cout << "Age: ";
  cin >> this->age;
  cout << "Gender (m/f): ";
  cin >> this->gender;

  cout << "Enter Marks for" << endl;

  for (i = 0; i < NSUBS; i++) {
    cout << "Subject " << i + 1 << ": ";
    cin >> this->marks[i];

    if (marks[i] > 100)
      exit(1);

    s += this->marks[i];
```

```
  }

  this->percentage = (double)s / NSUBS;
}

inline char *Student::getName() { return this->name; }
inline double Student::getPercentage() { return this->percentage; }

int main() {
  int i;

  Student a[NSTUDENTS];

  for (i = 0; i < NSTUDENTS; i++)
    a[i].input();

  cout << "\nStudents with percentage greater than 70:\n";

  for (i = 0; i < NSTUDENTS; i++)
    if (a[i].getPercentage() > 70)
      cout << a[i].getName() << endl;

  return 0;
}
```

```
Name: Amaang
Branch: InformationTechnology
Enter Enrollment number: 413
Age: 14
Gender (m/f): m
Enter Marks for
Subject 1: 99
Subject 2: 93
Subject 3: 9
Subject 4: 91
Subject 5: 3
Name: Allu
Branch: Physics
Enter Enrollment number: 444
Age: 19
Gender (m/f): f
Enter Marks for
Subject 1: 100
Subject 2: 0
Subject 3: 100
Subject 4: 0
Subject 5: 100
Name: Asli
Branch: Law
Enter Enrollment number: 33
Age: 99
Gender (m/f): m
Enter Marks for
```

```
Subject 1: 1
Subject 2: 1
Subject 3: 1
Subject 4: 1
Subject 5: 2
Name: how
Branch: Music
Enter Enrollment number: 9999
Age: 9
Gender (m/f): m
Enter Marks for
Subject 1: 12
Subject 2: 22
Subject 3: 32
Subject 4: 42
Subject 5: 52
Name: sabji
Branch: CivilEngineering
Enter Enrollment number: 91

Students with percentage greater than 70:
sabji
```

3 Lab 3

3.1 Write a program to enter any number and find its factorial using constructor.

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

uint64_t factorial(uint n) { return (n ? n * factorial(n - 1) : 1); }

class NumberWithFactorial {
private:
  uint n;
  uint64_t f;

public:
  NumberWithFactorial(uint);
};

NumberWithFactorial::NumberWithFactorial(uint n) {
  this->n = n;
  this->f = factorial(n);
  cout << "Factorial of " << this->n << " is " << this->f << endl;
}

int main() {
  NumberWithFactorial(5);
}
```

```
Factorial of 5 is 120
```

3.2 Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imag parts to equal values and third which takes two arguments is used to initialize real and imag to two different values.

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

class Complex {
private:
  int r, i;

public:
  Complex() = default;
  Complex(int);
  Complex(int, int);
  Complex operator+(Complex);
  void display();
};

Complex::Complex(int a) { this->r = this->i = a; }

Complex::Complex(int r, int i) {
  this->r = r;
  this->i = i;
}

Complex Complex::operator+(Complex op) {
  return Complex(this->r + op.r, this->i + op.i);
}

void Complex::display() {
  if (this->r)
    cout << this->r;

  if (this-> r && this->i > 0)
    cout << '+';

  if (this->i)
    cout << this->i << 'i';

  cout << endl;
}

int main() {
  Complex c;
  Complex a(4), b(34, -2124);
  c = a + b;
  c.display();
}
```

```
38-2120i
```

3.3 Write a program to generate a Fibonacci series using a copy constructor.

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

class Fibo {
private:
  uint lim;

public:
  Fibo() = default;
  Fibo(uint);
  Fibo(const Fibo &);
};

Fibo::Fibo(uint n) { this->lim = n; }

Fibo::Fibo(const Fibo &f) {
  uint a = 0, b = 1, i;

  cout << a << ' ' << b;

  for (i = 0; i < f.lim - 2; i++) {
    cout << ' ' << a + b;
    b += a;
    a = b - a;
  }
  cout << endl;
}

int main() {
  Fibo a(10);
  Fibo b = a;
}
```

```
0 1 1 2 3 5 8 13 21 34
```

4 Lab 4

4.1 Write a program to find the biggest of three numbers using friend function.

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

class Trio {
private:
  int a, b, c;

public:
  Trio(int, int, int);
  friend int biggest(Trio);
};

Trio::Trio(int a, int b, int c) {
  this->a = a;
  this->b = b;
  this->c = c;
}

int biggest(Trio t) { return (max({t.a, t.b, t.c})); }

int main() {
  Trio t(444, 4, -44);
  cout << biggest(t);
}
```

```
444
```

4.2 Write a program to demonstrate the use of friend function with Inline assignment.

All friend functions are inline functions.

4.3 Write a program to find the greatest of two given numbers in two different classes using friend function.

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

class Foo;
class Bar;

class Foo {
private:
  int foo;

public:
  Foo(int n) { foo = n; }
  friend int bigger(Foo, Bar);
};

class Bar {
private:
  int bar;

public:
  Bar(int n) { bar = n; }
  friend int bigger(Foo, Bar);
};

int bigger(Foo a, Bar b) { return max(a.foo, b.bar); }

int main() {
  Foo a(5);
  Bar b(55);

  cout << bigger(a, b);
}
```

```
55
```

5 Lab 5

5.1 Imagine a publishing company that markets both book and audiocassette versions of its works. Create a class publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int), and tape, which adds a playing time in minutes (type float). Each of these three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata() function to display its data. (Simple inheritance & method overriding)

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

class Work {
private:
  char title[50];
  uint price;

public:
  void getData();
  void putData();
};

void Work::getData() {
  cout << "Title of publication: ";
  cin >> title;
  cout << "Price of publication: ";
  cin >> price;
}

void Work::putData() {
  cout << "Title of publication: " << title << endl
       << "Price of publication: " << price << endl;
}

class Book : public Work {
private:
  uint pageCount;

public:
  void getData();
  void putData();
};

void Book::getData() {
  Work::getData();
  cout << "Number of pages: ";
  cin >> pageCount;
}

void Book::putData() {
  Work::putData();
  cout << "Number of pages: " << pageCount << endl;
}
```

```
class Tape : public Work {
private:
  uint lengthMin;

public:
  void getData();
  void putData();
};

void Tape::getData() {
  Work::getData();
  cout << "Length in minutes: ";
  cin >> lengthMin;
}

void Tape::putData() {
  Work::putData();
  cout << "Length in minutes: " << lengthMin << endl;
}

int main() {
  Tape t;
  Book b;
  t.getData();
  t.putData();
  b.getData();
  b.putData();
}
```

```
Title of publication: alo
Price of publication: 4
Length in minutes: 4
Title of publication: alo
Price of publication: 4
Length in minutes: 4
Title of publication: cope
Price of publication: 44
Number of pages: 444
Title of publication: cope
Price of publication: 44
Number of pages: 444
```

5.2 C++ program to read and print employee information using multiple inheritance. The program has following classes:

5.2.1 basicInfo

- Data: name[char,30],empId[int], gender[char]
- Functions: getData(), putData();

5.2.2 deptInfo

- Data: deptName[char,30], assignWork[char,30], timeToComplete(int)
- Functions: getData(), putData();

5.2.3 employeeInfo

- Data: salary[int], age[int]
- Functions: getData(), putData();

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

class BasicInfo {
private:
  char name[30];
  int empId;
  char gender;

public:
  void getData();
  void putData();
};

void BasicInfo::getData() {
  cout << "Name: ";
  cin >> name;
  cout << "Employee ID: ";
  cin >> empId;
  cout << "Gender: ";
  cin >> gender;
}

void BasicInfo::putData() {
  cout << "Name: " << name << endl
       << "Employee ID: " << empId << endl
       << "Gender: " << gender << endl;
}

class DeptInfo : public BasicInfo {
private:
  char deptName[30];
  char assignWork[30];
  int timeToComplete;
```

```
public:
  void getData();
  void putData();
};

void DeptInfo::getData() {
  BasicInfo::getData();
  cout << "Name of Department: ";
  cin >> deptName;
  cout << "Assigned Work: ";
  cin >> assignWork;
  cout << "Time to complete: ";
  cin >> timeToComplete;
}

void DeptInfo::putData() {
  BasicInfo::putData();
  cout << "Name of Department: " << deptName << endl
       << "Assigned Work: " << assignWork << endl
       << "Time to complete: " << timeToComplete << endl;
}

class EmployeeInfo : public DeptInfo {
private:
  int salary;
  int age;

public:
  void getData();
  void putData();
};

void EmployeeInfo::getData() {
  DeptInfo::getData();
  cout << "Salary of employee: ";
  cin >> salary;
  cout << "Age of employee: ";
  cin >> age;
}

void EmployeeInfo::putData() {
  DeptInfo::putData();
  cout << "Salary of employee: " << salary << endl
       << "Age of employee: " << age << endl;
}

int main() {
  EmployeeInfo e;
  e.getData();
  e.putData();

  return 0;
```

```
}
```

```
Name: QuantaviousQuandaleIII
Employee ID: 4
Gender: f
Name of Department: BasedDepartment
Assigned Work: DiscordModeration
Time to complete: 4
Salary of employee: 444444
Age of employee: 44
Name: QuantaviousQuandaleIII
Employee ID: 4
Gender: f
Name of Department: BasedDepartment
Assigned Work: DiscordModeration
Time to complete: 4
Salary of employee: 444444
Age of employee: 44
```

5.3 Design three classes STUDENT ,EXAM and RESULT. The STUDENT class has data members such as rollno, name. create a class EXAM by inheriting the STUDENT class. The EXAM class adds data members representing the marks scored in six subjects. Derive the RESULT from the EXAM class and has its own data members such as total marks. Write a program to model this relationship.

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

class Student {
private:
  char name[30];
  int enrollment;

public:
  void getData();
};

void Student::getData() {
  cout << "Name: ";
  cin >> name;
  cout << "Enrollment: ";
  cin >> enrollment;
}

class Exam : public Student {
protected:
  const static uint NSUBS = 6;
  uint marks[NSUBS];

public:
  void getData();
};

void Exam::getData() {
  uint i;

  Student::getData();

  cout << "Enter marks for " << endl;

  for (i = 0; i < NSUBS; i++) {
    cout << "Subject " << i + 1 << ": ";
    cin >> marks[i];
  }
}

class Result : private Exam {
private:
  uint totalMarks;

public:
  void getData();
```

```
  uint getTotal();
};

void Result::getData() {
  uint i;

  Exam::getData();
  this->totalMarks = 0;

  for (i = 0; i < Exam::NSUBS; i++)
    this->totalMarks += Exam::marks[i];
}

uint Result::getTotal() { return this->totalMarks; }

int main() {
  Result r;
  r.getData();
  cout << "Total Marks: " << r.getTotal();
  return 0;
}
```

```
Name: RetardSingh
Enrollment: 4
Enter marks for
Subject 1: 44
Subject 2: 4
Subject 3: 30
Subject 4: 99
Subject 5: 26
Subject 6: 84
```

6 Lab 6

6.1 Create class first with data members book no, book name and member function getdata and putdata. Create a class second with data members author name ,publisher and members getdata and showdata. Derive a class third from first and second with data member no of pages and year of publication. Display all these information using an array of objects of third class.

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

class Book {
private:
  char name[50];
  uint id;

public:
  void getData();
  void putData();
};

void Book::getData() {
  cout << "Name of book: ";
  cin >> name;
  cout << "ID of book: ";
  cin >> id;
}

void Book::putData() {
  cout << "Name of book: " << name << endl << "ID of book: " << id << endl;
}

class Authorities : public Book {
private:
  char author[50];
  char publisher[50];

public:
  void getData();
  void putData();
};

void Authorities::getData() {
  cout << "Name of author: ";
  cin >> author;
  cout << "Name of publisher: ";
  cin >> publisher;
}

void Authorities::putData() {
  cout << "Name of author: " << author << endl
       << "Name of publisher: " << publisher << endl;
}
```

```
class Publication : public Authorities {
private:
  uint pageCount;
  uint year;

public:
  void getData();
  void putData();
};

void Publication::getData() {
  cout << "Number of pages: ";
  cin >> pageCount;
  cout << "Year (YYYY): ";
  cin >> year;
}

void Publication::putData() {
  cout << "Number of pages: " << pageCount << endl << "Year: " << year << endl;
}

int main() {
  Publication p[3];

  for (auto &x: p) {
    x.Book::getData();
    x.Authorities::getData();
    x.getData();
  }

  for (auto &x: p) {
    x.Book::putData();
    x.Authorities::putData();
    x.putData();
  }
}
```

```
ID of book: 4
Name of author: a.
Name of publisher: a.
Number of pages: 4
Year (YYYY): 4044
Name of book: sahi
ID of book: 5
Name of author: baat
Name of publisher: hai
Number of pages: 555555
Year (YYYY): 192
Name of book: how
ID of book: 349234
Name of author: ok
Name of publisher: test
```

```
Number of pages: 99991293
Year (YYYY): 2022
Name of book: Allu
ID of book: 4
Name of author: a.
Name of publisher: a.
Number of pages: 4
Year: 4044
Name of book: sahi
ID of book: 5
Name of author: baat
Name of publisher: hai
Number of pages: 555555
Year: 192
Name of book: how
ID of book: 349234
Name of author: ok
Name of publisher: test
Number of pages: 99991293
Year: 2022
```

6.2 Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRIANGLE or RECTANGLE interactively and display the area.

```
#include <iostream>
#include <math.h>
using namespace std;

class Shape {
  int stub;

public:
  virtual void getData() { cout << "stub"; }

  virtual double area() {
    cout << "stub";
    return 0;
  }
};

class Triangle : public Shape {
private:
  double sides[3];

public:
  void getData();
  double area();
};

void Triangle::getData() {
  cout << "Enter 3 integers for the sides of triangle: \n";

  for (int i = 0; i < 3; i++)
    cin >> sides[i];
}

double Triangle::area() {
  double s = (sides[0] + sides[1] + sides[2]) / 2;

  return sqrt(s * (s - sides[0]) * (s - sides[1]) * (s - sides[2]));
}

class Rectangle : public Shape {
private:
  double l;
  double b;

public:
  void getData();
```

```
  double area();
};

void Rectangle::getData() {
  cout << "Enter length of rectangle: ";
  cin >> l;
  cout << "Enter breadh of rectangle: ";
  cin >> b;
}

double Rectangle::area() { return l * b; }

int main() {
  Shape *s;
  Triangle t;
  Rectangle r;

  s = &t;

  s->getData();
  cout << "Area of the Shape: " << s->area() << endl;

  s = &r;

  s->getData();
  cout << "Area of the Shape: " << s->area() << endl;

}
```

```
Enter 3 integers for the sides of triangle:
4
4
4
Area of the Shape: 6.9282
Enter length of rectangle: 4
Enter breadh of rectangle: 4
Area of the Shape: 16
```

6.3 Create a base class basic_info with data members name ,roll no, gender and two member functions getdata and display. Derive a class physical_fit from basic_info which has data members height and weight and member functions getdata and display. Display all the information using object of derived class.

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

class BasicInfo {
private:
  char name[30];
  uint enrollment;
  char gender;

public:
  void getData();
  void putData();
};

void BasicInfo::getData() {
  cout << "Name: ";
  cin >> name;
  cout << "Enrollment: ";
  cin >> enrollment;
  cout << "Gender: ";
  cin >> gender;
}
void BasicInfo::putData() {
  cout << "Name: " << name << endl
       << "Enrollment: " << enrollment << endl
       << "Gender: " << gender << endl;
}

class PhysicalFit : public BasicInfo {
  uint height;
  uint weight;

public:
  void getData();
  void putData();
};

void PhysicalFit::getData() {
  cout << "Weight (in kg): ";
  cin >> weight;
  cout << "Height (in cm): ";
  cin >> height;
}

void PhysicalFit::putData() {
  cout << "Weight (in kg): " << weight << endl
       << "Height (in cm): " << height << endl;
}
```

```
int main() {
  PhysicalFit x;
  x.BasicInfo::getData();
  x.getData();

  x.BasicInfo::putData();
  x.putData();

  return 0;
}
```

```
Name: acha
Enrollment: 4
Gender: f
Weight (in kg): 999
Height (in cm): 2
Name: acha
Enrollment: 4
Gender: f
Weight (in kg): 999
Height (in cm): 2
```