(Лабораторн №7)

Э.Багабанди

ХШУИС - ийн ПХ хөтөлбөрийн

3-р түвшний оюутан, 19B1NUM0700@num.edu.mn

1. **ОРШИЛ/УДИРТГАЛ**

Уг лабораторийн хүрээнд өгөгдсөн дасгалуудыг шийдвэрлэж, тайлбарлахад ашиглагдсан C++ хэлний онолын ойлголт /удамшил/ - уудыг судалж, эзэмшсэн болно.

2. **ЗОРИЛГО**

Энэхүү лабораторийн ажлын хүрээнд C++ хэлний удамшил, түүний горим/public, private, protected/, давуу тал болон төрлүүдийн талаар судалж, үүнийг ашиглан лабораторт өгөгдсөн дасгалуудыг гүйцэтгэн жижиг хэмжээний хэрэгжүүлэлт хийх юм.

3. **ОНОЛЫН СУДАЛГАА**

3.1. Удамшил гэж юу вэ? (тодорхойлолт, C++ дээр хэрхэн хэрэгжүүлдэг талаар бичнэ)

- Удамшил гэдэг нь нэгэнт бий болсон зүйлийг дахин хэрэглэхтэй холбоотой ойлголт бөгөөд туршиж бэлэн болгосон классын кодыг өөр системд дахин хэрэглэх буюу эргэн хэрэглэгдэх чадвар, чадамжтай болгодог. Өөрөөр хэлбэл бэлэн байгаа классаас шинэ класс үүсгэх арга технологи гэж хэлж болох бөгөөд энэ үед шинээр үүсч буй класс нь өмнөх классынхаа шинжийг өвлөн авна.
- Өөрөөс нь класс үүсч байгаа классыг үндсэн(дээд/эх) класс, шинээр үүсч байгааг нь удамших, удамшсан(дэд/охин/хүүхэд) класс гэнэ.
- Удамшиж үүсэх класс нь эх классынхаа шинж, аргаас өвлөж авахаас гадна өөрийн гэсэн шинжтэй, өөрөөр хэлбэл эх класст байхгүй гишүүдтэй байж болно. Эх классын гишүүн функцийг удамших класс дотор дахин тодорхойлж болох ба ингэхэд эх классын эх код заавал байх албагүй юм.

3.2. Удамшлын горим. public, private, protected горимын талаар тайлбарлаж жишээгээр батална.

- **Классыг <u>public</u> горимоор удамшуулж үүсгэх** үеийн удамших классын тодорхойлолт:

```
class derived_class: public based_class
{
        data_members;
        members_function;
}
```

- Эх класс <u>public</u> горимоор удамших тохиолдолд түүний <u>public</u> гишүүд удамшиж үүсэх классын <u>public</u> гишүүд болно. Иймээс удамших классын функц ба объект эх классынхаа public гишүүд рүү хандана. Харин эх класс <u>public</u> горимоор удамших үед түүний <u>private</u> гишүүд удамших хэдий ч удамших классын гишүүд эх классынхаа private гишүүд рүү хандаж чадахгүй. Мөн эх класс <u>public</u> горимоор удамших үед

түүний protected гишүүд удамших классынхаа protected гишүүд болох тул уг гишүүд рүү удамших классын функцээс хандаж болно.

- Тухайлбал,

```
class BB{
private:
        int aa;
protected:
        int bb;
public:
        int cc;
------------------------
------------------------
};
```

BB эх классаас DD классыг public горимоор удамшуулж үүсгэх үед:

```
class DD: public BB{
-------------------------
-------------------------
};
```

DD класс нь доор үзүүлсэн бүтэцтэй болох бөгөөд цаашдаа өөр классын эх класс болж чадна.

```
Class DD{
protected:
        int bb;
public:
        int cc;
------------------------
------------------------
};
```

- **Классыг private горимоор удамшуулж үүсгэх** үеийн удамших классын тодорхойлолт:

```
class derived_class: private based_class
{
        data_members;
        members_function;
}
```

- Эх класс private горимоор удамших тохиолдолд түүний public болон protected гишүүд удамшиж үүсэх классын private гишүүд болно. Иймд удамших классын объект эх классынхаа гишүүд рүү хандаж чадахгүй. Харин функц нь классынхаа private гишүүд рүү хандаж болдог тул эх классынхаа public, protected гишүүд рүү удамших классын функцээс хандана.
- Тухайлбал,

```
class BB{
private:
        int aa;
protected:
        int bb;
public:
        int cc;
------------------------
------------------------
};
```

BB эх классаас DD классыг private горимоор удамшуулж үүсгэх үед:

```
class DD: public BB{
-------------------------
-------------------------
};
```

DD класс нь доор үзүүлсэн бүтэцтэй болох бөгөөд цаашдаа өөр классын эх класс болж чадна.

```
Class DD{
private:
        int bb;
        int cc;
------------------------
------------------------
};
```

- **Классыг <u>protected</u> горимоор удамшуулж үүсгэх** үеийн удамших классын тодорхойлолт:

```
class derived_class: protected based_class
{
        data_members;
        members_function;
}
```

- Эх класс protected горимоор удамших тохиолдолд түүний public болон protected гишүүд удамшиж үүсэх классын protected гишүүд болно. Иймд уг гишүүд рүү удамших классын функцээс хандаж болно. Харин эх класс <u>protected</u> горимоор удамших үед түүний <u>private</u> гишүүд удамших хэдий ч удамших классын гишүүд эх классынхаа private гишүүд рүү хандаж чадахгүй.
- <u>Жишээ код:</u>

```
//Finding gross pay
class Employee{
protected:
	char name[20];
	int basicpay;
	int allowance;
	int grosspay;
public:
	void getData(){
		cout<<"\nEnter the employee Name: ";
		gets(name);
		cout<<"Enter the basic pay: ";
		cin>>basicpay;
		cout<<"Enter the allowance: ";
		cin>>allowance;
	}
	void showData(){
		cout<<endl;
		cout<<name<<endl;
		cout<<basicpay<<endl;
		cout<<allowance<<endl;
	}
};
class Engineer:public Employee{
private:
	int bonus;
public:
	void getData(){
		Employee::getData();
		cout<<"Enter the bonus: ";
		cin>>bonus;
	}
	void showData(){
		grosspay = basicpay + allowance + bonus;
		Employee::showData();
		cout<<bonus<<endl;
		cout<<grosspay<<endl;
	}
}
int main(){
	Engineer en;
	en.getData();
	en.showData();
	return 0;
}
```
[1]

[1] Жишээ код №01

```
Enter the employee Name: Bagaa
Enter the basic pay: 9000
Enter the allowance: 8000
Enter the bonus: 3000

Name: Bagaa
Basicpay: 9000
Allowance: 8000
Bonus: 3000
Grosspay: 20000
```

- Удамших горимын хүснэгт:

Эх классын гишүүд	Удамших горим		
	public	private	protected
public	public	private	protected
private	X	X	X
protected	protected	private	protected

3.3. Удамшлын давуу талуудыг тоочин бичиж бодит жишээн дээр тайлбарла.

- Програмчлалд удамших шинжийг хэрэглэснээр дараах боломжууд бүрдэнэ.
  - Код болон програмыг **дахин хэрэглэх** – Жишээ код №1 дээрх Employee классыг удамшуулан Engineer классыг үүсгэсэн бөгөөд ингэснээр Employee классын кодыг дахин бичихгүйгээр шууд Engineer класс дотор хэрэглэх боломжтой болсон.
  - Код болон програмыг кодыг **дундаа хэрэглэх** – Жишээ код №1 дээрх Employee классын гишүүн өгөгдөл болон функцийг Engineer класс хандах эрхээс хамааран дундаа хэрэглэх боломжтой байгаа нь харагдаж байна.
  - Код болон програмыг **тогтмол интерфейс хэрэглэх** - Интерфейс буюу гишүүн функцийг цуглуулга бүхий классыг үүсгэн удамшуулж, дэд классуудад дахин тодорхойлон ашиглах боломжтой. Жишээ код №1 дээр Employee классын гишүүн функцийг Engineer класс дахин тодорхойлж ашигласантай ижил.

3.4. Удамшлын хэдэн төрөл байдаг вэ? Тус бүрийг тайлбарлан бич.

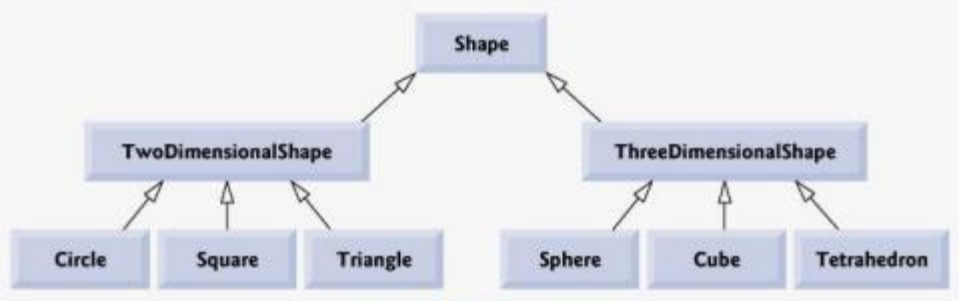
- Удамшлын 5 төрөл байдаг. Үүнд:
  - Нэг – нэг буюу дан, энгийн удамшил. Удамших класс зөвхөн нэг эх класстай байна. Тухайлбал, үхэр нь амьтны аймагт багтах тул cow класс нь animals классаас удамшиж үүссэн гэж үзэж болно.
  - Олон – нэг буюу нийлмэл удамшил. Зарим тохиолдолд удамших буюу дэд класс нь 2 – оос цөөнгүй эх класстай байж болно. Тухайлбал, ачаа ачдаг, мөн олон 4 буюу түүнээс дээш тооны суудалтай машинуудыг суудлын машин болон ачааны машинаас удамшиж үүссэн гэж үзэж болно.
  - Олон түвшинт удамшил. Удамших класс өөрөө удамших классаас үүснэ. Тухайлбал, Z класс Y классаас, Y класс нь харин X классаас үүсэх бол энэ шинжийг олон түвшинт удамшил гэнэ.
  - Шаталсан удамшил. Нэг эх классаас 2-оос цөөнгүй удамших класс үүсэх бол ийм удамшлыг шаталсан удамшил гэнэ. Тухайлбал, МУИС – аас салбар сургуулиуд буюу Ахисан түвшний сургууль, Бизнесийн сургууль, Олон улсын харилцаа, нийтийн удирдлагын сургууль, Хууль зүйн сургууль, Хэрэглээний шинжлэх ухаан,

инженерчлэлийн сургууль, Шинжлэх ухааны сургууль, Завхан сургууль, Эрдэнэт сургууль, Байгаль-Эх Лицей Ахлах Сургууль, Экологийн Боловсролын Төв эдгээр нь шаталсан удамшлаар удамшсан гэж үзэж болох юм.

- Холимог удамшил. Олон – нэг буюу нийлмэл удамшил, шаталсан удамшил хоёрын нийлэмж удамшил юм. Тухайлбал, эвэр туурайтны аймагт багтах гүү, илжиг хоёрын дундаас гарах луус бол уг удамшлаар үүссэн гэж үзэж болно.

4. ХЭРЭГЖҮҮЛЭЛТ

Зураг 1-д дүрсэлсэн удамшлын модны дагуу хоёр хэмжээст геометрийн дүрсүүдийн классыг байгуул.



Зураг 1. Дүрсийн удамшил

Тойрог:

- Гишүүн өгөгдөл-тойргийн төвийн координат, радиус, нэр;
- Гишүүн функц-талбай олох, тойргийн урт тооцох
- Тойргийн төвийн цэг, радиусд утга онооно.

```
class Circle{
private:
   char* name;
   Point cp;
   float r;
public:
   Circle(){
      name = new char(1);
      strcpy(name, "");
      r = 1;
   }
   Circle(Point ct, float r, char *ner = ""){
      cp = ct;
      this->r = r;
      name = new char(strlen(ner) + 1);
      strcpy(name, ner);
   }
   ~Circle(){
      delete name;
   }
   float findArea(){
      return r*r*pi;
   }
   float findPerimeter(){
      return 2*r*pi;
   }
   void setCenterPoint(Point ct){
      cp = ct;
   }
   void setRadius(float radius){
      r = radius;
   }
   void getData(){
      cp<<"CP";
      cout<<"Name: "<<name<<endl;
      cout<<"Radius: "<<r<<endl;
   }
};
```

Квадрат:

- Гишүүн өгөгдөл-оройнуудын координат, талын урт, нэр;
- Гишүүн функц-талбай олох, приметр тооцох
- Квадратын талын урт, зүүн дээд оройн координатад утга оноогоод бусад оройг тооцоолж классын харгалзах гишүүн өгөгдөлд хадгална.

```
class Square{
private:
   Point A;
   Point B;
   Point C;
   Point D;
   float a;
   char*name;
   void calcOtherPoints(Point lt_p, float side){
      B.x = lt_p.x + side;
      B.y = lt_p.y;
      C.x = lt_p.x + side;
      C.y = lt_p.y - side;
      D.x = lt_p.x;
      D.y = lt_p.y - side;
   }
public:
   Square(){
      name = new char(1);
      strcpy(name, "");
      A.x = -1;
      A.y = 1;
      a = 2;
      calcOtherPoints(A, a);
   }
   Square(Point lt_p, float a, char *name = ""){
      this->name = new char(strlen(name) + 1);
      strcpy(this->name, name);
      this->a = a;
      setLTpoint(lt_p);
   }
   ~Square(){
      delete name;
   }
   float findArea(){
      return a*a;
   }
   float findPerimeter(){
      return 4*a;
   }
   void setAside(float a){
      this->a = a;
      calcOtherPoints(A, a);
   }
   void setLTpoint(Point lt_p){
      A = lt_p;
      calcOtherPoints(A, a);
   }

   void getData(){
      A<<"A";
      B<<"B";
      C<<"C";
      D<<"D";
      cout<<"Name: "<<name<<endl;
      cout<<"A side: "<<a<<endl;
   }
};
```

Зөв гурвалжин:

- Гишүүн өгөгдөл-оройнуудын координат, талуудын урт, нэр;
- Гишүүн функц-талбай, приметр тооцох
- Зөв гурвалжингийн талын урт, дээд оройн координатыг утга оноогоод бусад оройг тооцоолно

```
class RightTriangle{
private:
   Point A;
   Point B;
   Point C;
   float a;
   char* name;
   //BC taliig Ox tenkhlegtei parallel gj uzsen bolno.
   void calcOtherPoints(Point tp, float side){
      B.x = tp.x + side/2;
      B.y = tp.y - sqrt(3)/2*side;
      C.x = tp.x - a/2;
      C.y = tp.y - sqrt(3)/2*side;
   }
public:
   RightTriangle(){
      name = new char(1);
      strcpy(name, "");
      A.x = 0;
      A.y = 1;
      a = 3;
      calcOtherPoints(A, a);
   }
   RightTriangle(Point pt, float a, char *name = ""){
      this->name = new char(strlen(name) + 1);
      strcpy(this->name, name);
      this->a = a;
      setTpoint(pt);
   }
   ~RightTriangle(){
      delete name;
   }
   float findArea(){
      return sqrt(3)*a*a/4;
   }
   float findPerimeter(){
      return 3*a/2;
   }
   void setAside(float a){
      this->a = a;
      calcOtherPoints(A, a);
   }
   void setTpoint(Point pt){
      A = pt;
      calcOtherPoints(A, a);
   }
   void getData(){
      A<<"A";
      B<<"B";
      C<<"C";
      cout<<"Name: "<<name<<endl;
      cout<<"A side: "<<a<<endl;
   }
};
```

Классуудад байгаа нийтлэг шинжийг эх классад оруулаад түүнээсээ удамшуулж классыг үүсгэнэ.

```
class Shape{
protected:
   char* name;
   Point startVertex;
   float side;
public:
   Shape(){
      name = new char(1);
      strcpy(name, "");
      float side = 1;
   }
   Shape(Point vertex, float a, char *name = ""){
      this->name = new char(strlen(name) + 1);
      strcpy(this->name, name);
      startVertex = vertex;
      side = a;
   }
   ~Shape(){
      delete name;
   }
   char* getName(){
      return name;
   }
   float getAside(){
      return side;
   }
   void setAside(float a){
      side = a;
   }
   Point* getVertex(){
      return &startVertex;
   }
   void setVertex(Point pt){
      startVertex = pt;
   }
};
class TwoDimensionalShape:public Shape{
public:
   TwoDimensionalShape(): Shape(){}
   TwoDimensionalShape(Point vertex, float a, char*name = ""): Shape(vertex, a, name){}
   float findArea();
   float findPerimeter();
};

class Circle:public TwoDimensionalShape{
public:
   Circle():TwoDimensionalShape(){}
   Circle(Point ct, float r, char *ner = ""):TwoDimensionalShape(ct, r, ner){}
   float findArea(){
      return side*side*pi;
   }
   float findPerimeter(){
      return 2*side*pi;
   }
   void setCenterPoint(Point ct){
      startVertex = ct;
   }
   void setRadius(float radius){
      side = radius;
   }
   void getData(){
```

```
startVertex<<"CP";
        cout<<"Name: "<<name<<endl;
        cout<<"Radius: "<<side<<endl;
    }
};
class Square:public TwoDimensionalShape{
private:
    Point B;
    Point C;
    Point D;
    void calcOtherPoints(Point lt_p, float side){
      B.x = lt_p.x + side;
      B.y = lt_p.y;
      C.x = lt_p.x + side;
      C.y = lt_p.y - side;
      D.x = lt_p.x;
      D.y = lt_p.y - side;
    }
public:
    Square():TwoDimensionalShape(){
        Point A(-1, 1);
        float a = 2;

        startVertex = A;
        side = a;
        calcOtherPoints(startVertex, side);
    }
    Square(Point lt_p, float a, char *name = ""):TwoDimensionalShape(lt_p, a, name){
        setLTpoint(lt_p);
    }
    float findArea(){
        return side*side;
    }
    float findPerimeter(){
        return 4*side;
    }
    void setAside(float a){
        side = a;
        calcOtherPoints(startVertex, side);
    }
    void setLTpoint(Point lt_p){
        startVertex = lt_p;
        calcOtherPoints(startVertex, side);
    }
    void getData(){
        startVertex<<"A";
        B<<"B";
        C<<"C";
        D<<"D";
        cout<<"Name: "<<name<<endl;
        cout<<"A side: "<<side<<endl;
    }
};
class RightTriangle:public TwoDimensionalShape{
private:
    Point B;
    Point C;
    //BC taliig Ox tenkhlegtei parallel gj uzsen bolno.
    void calcOtherPoints(Point tp, float a){
        B.x = tp.x + side/2;
        B.y = tp.y - sqrt(3)/2*side;
```

```
        C.x = tp.x - a/2;
        C.y = tp.y - sqrt(3)/2*side;
    }
public:
    RightTriangle(): TwoDimensionalShape(){
        Point A(0, 1);
        startVertex = A;
        float a = 3;
        side = a;
        calcOtherPoints(startVertex, side);
    }
    RightTriangle(Point pt, float a, char *name = ""):TwoDimensionalShape(pt, a, name){
        setTpoint(pt);
    }
    float findArea(){
        return sqrt(3)*side*side/4;
    }
    float findPerimeter(){
        return 3*side/2;
    }
    void setAside(float a){
        side = a;
        calcOtherPoints(startVertex, side);
    }
    void setTpoint(Point pt){
        startVertex = pt;
        calcOtherPoints(startVertex, side);
    }
    void getData(){
        startVertex<<"A";
        B<<"B";
        C<<"C";
        cout<<"Name: "<<name<<endl;
        cout<<"A side: "<<side<<endl;
    }
};
```

5. **ДҮГНЭЛТ**

Энэхүү лабораторын ажлын хүрээнд C++ хэл дээр жижиг хэмжээний програм хөгжүүлсэн бөгөөд тухайн програмыг хэрэгжүүлэхдээ олон түвшинт болон шаталсан төрлийн удамшлыг үүсгэж, public горимоор удамшуулж, эх классуудыг гишүүн өгөгдлийг protected – оор тодорхойлсон. Үүнээс удамших классууд эх классуудынхаа public болон protected гишүүн өгөгдөл болон дүрмүүдэд функц дотроос хандах боломжтой болох юм. Мөн классуудыг удамшуулан үүсгэхийн өмнө, дан дангаар үүсгэн класс бүрт хадгалагдаж байгаа нийтлэг шинж, дүрмүүдийг авч эх классуудыг үүсгэсэн болно. Ингэснээр удамшлын давуу талыг бий болж, дахин хэрэглэгдэх өндөр чадвар, чадамжтай класс үүсгэх боломж бүрдэх юм.

6. **АШИГЛАСАН МАТЕРИАЛ**

1. Объект хандлагат технологийн C++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.

7. **ХАВСРАЛ**

```cpp
#include <iostream>
#include <string.h>
#include <math.h>
#define pi 3.1416
using namespace std;
class Point {
public:
    float x, y;
    Point() {
        x = 0;
        y = 0;
    }

    Point(int x, int y){
        this->x = x;
        this->y = y;
    }

    void operator<<(char s[]){
        cout<< s << "(" << this->x << ", " << this->y << ")" << endl;
    }
};
class Shape{
protected:
    char* name;
    Point startVertex;
    float side;
public:
    Shape(){
        name = new char(1);
        strcpy(name, "");
        float side = 1;
    }
    Shape(Point vertex, float a, char *name = ""){
        this->name = new char(strlen(name) + 1);
        strcpy(this->name, name);
        startVertex = vertex;
        side = a;
    }
    ~Shape(){
        delete name;
    }
    char* getName(){
        return name;
    }
    float getAside(){
        return side;
    }
    void setAside(float a){
        side = a;
    }
    Point* getVertex(){
        return &startVertex;
    }
    void setVertex(Point pt){
        startVertex = pt;
    }
};
class TwoDimensionalShape:public Shape{
public:
    TwoDimensionalShape(): Shape(){}
    TwoDimensionalShape(Point vertex, float a, char*name = ""): Shape(vertex, a, name){}
    float findArea();
    float findPerimeter();
};

class Circle:public TwoDimensionalShape{
public:
    Circle():TwoDimensionalShape(){}
    Circle(Point ct, float r, char *ner = ""):TwoDimensionalShape(ct, r, ner){}
    float findArea(){
        return side*side*pi;
    }
    float findPerimeter(){
        return 2*side*pi;
    }
    void setCenterPoint(Point ct){
        startVertex = ct;
    }
    void setRadius(float radius){
        side = radius;
    }
    void getData(){
        startVertex<<"CP";
```

```
85          cout<<"Name: "<<name<<endl;
86          cout<<"Radius: "<<side<<endl;
87      }
88  };
89  class Square:public TwoDimensionalShape{
90  private:
91      Point B;
92      Point C;
93      Point D;
94      void calcOtherPoints(Point lt_p, float side){
95         B.x = lt_p.x + side;
96         B.y = lt_p.y;
97         C.x = lt_p.x + side;
98         C.y = lt_p.y - side;
99         D.x = lt_p.x;
100        D.y = lt_p.y - side;
101     }
102 public:
103     Square():TwoDimensionalShape(){
104         Point A(-1, 1);
105         float a = 2;
106
107         startVertex = A;
108         side = a;
109         calcOtherPoints(startVertex, side);
110     }
111     Square(Point lt_p, float a, char *name = ""):TwoDimensionalShape(lt_p, a, name){
112         setLTpoint(lt_p);
113     }
114     float findArea(){
115         return side*side;
116     }
117     float findPerimeter(){
118         return 4*side;
119     }
120     void setAside(float a){
121         side = a;
122         calcOtherPoints(startVertex, side);
123     }
124     void setLTpoint(Point lt_p){
125         startVertex = lt_p;
126         calcOtherPoints(startVertex, side);
127     }
128     void getData(){
129         startVertex<<"A";
130         B<<"B";
131         C<<"C";
132         D<<"D";
133         cout<<"Name: "<<name<<endl;
134         cout<<"A side: "<<side<<endl;
135     }
136 };
137 class RightTriangle:public TwoDimensionalShape{
138 private:
139     Point B;
140     Point C;
141     //BC талыг Ox тэнхлэгтэй parallel ги үзсэн болно.
142     void calcOtherPoints(Point tp, float a){
143         B.x = tp.x + side/2;
144         B.y = tp.y - sqrt(3)/2*side;
145         C.x = tp.x - a/2;
146         C.y = tp.y - sqrt(3)/2*side;
147     }
148 public:
149     RightTriangle(): TwoDimensionalShape(){
150         Point A(0, 1);
151         startVertex = A;
152         float a = 3;
153         side = a;
154         calcOtherPoints(startVertex, side);
155     }
156     RightTriangle(Point pt, float a, char *name = ""):TwoDimensionalShape(pt, a, name){
157         setTpoint(pt);
158     }
159     float findArea(){
160         return sqrt(3)*side*side/4;
161     }
162     float findPerimeter(){
163         return 3*side/2;
164     }
165     void setAside(float a){
166         side = a;
167         calcOtherPoints(startVertex, side);
168     }
```

```
169         void setTpoint(Point pt){
170             startVertex = pt;
171             calcOtherPoints(startVertex, side);
172         }
173         void getData(){
174             startVertex<<"A";
175             B<<"B";
176             C<<"C";
177             cout<<"Name: "<<name<<endl;
178             cout<<"A side: "<<side<<endl;
179         }
180     };
181
182     int main(){
183         Point ct(1, 1), st(-2, 2), tt(2, 2);
184
185         Circle c1(ct, 3, "C1");
186         cout<<"Circle: "<<endl;
187         c1.getData();
188         cout<<"Area: "<<c1.findArea()<<endl;
189         cout<<"Length: "<<c1.findPerimeter();
190         cout<<"\n\n";
191
192         Square s1(st, 2, "S1");
193         cout<<"Square: "<<endl;
194         s1.getData();
195         cout<<"Area: "<<s1.findArea()<<endl;
196         cout<<"Length: "<<s1.findPerimeter();
197         cout<<"\n\n";
198
199         RightTriangle rt1(tt, 2, "RT1");
200         cout<<"RightTrangle: "<<endl;
201         rt1.getData();
202         cout<<"Area: "<<rt1.findArea()<<endl;
203         cout<<"Length: "<<rt1.findPerimeter();
204         cout<<"\n\n";
205
206         return 0;
207     }
208
```