(Лабораторн №9)

Э.Багабанди

ХШУИС - ийн ПХ хөтөлбөрийн

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

1. ОРШИЛ/УДИРТГАЛ

Уг лабораторийн хүрээнд өгөгдсөн дасгалуудыг гүйцэтгэн 7-аар лабораторт хөгжүүлсэн жижиг хэмжээний програмыг өргөтгөн хөгжүүлсэн бөгөөд үүнд ашиглагдсан C++ хэлний онолын ойлголт /жинхэнэ хийсвэр класс, жинхэнэ хийсвэр функц, хийсвэр функц, функц дахин программчлах, удамшил ба устгагч функцын хамаарал/ - уудыг судалж, эзэмшсэн.

2. ЗОРИЛГО

C++ хэлний хийсвэр функц, жинхэнэ хийсвэр функц, функц дахин программчлах, хийсвэр класс, удамшил ба устгагч функцын хамааралын талаар судалж, үүнийг ашиглан лабораторт өгөгдсөн дасгалуудыг гүйцэтгэн жижиг хэмжээний хэрэгжүүлэлт хийх.

3. ОНОЛЫН СУДАЛГАА

3.1. Хийсвэр функц гэж юу вэ? Тодорхойлолт, зарлалт, жишээ, давуу талыг тус тус бич.

- *Тодорхойлолт:* Хийсвэр функц нь эх классийн гишүүн функц бөгөөд удамшуулах үед түүнийг дахин програмчилж ашиглах боломжтой функцийг хэлэх ба тухайн эх классд тодорхойлогдсон хийсвэр аргын хэрэгжилтийг хүү классууд нь өөр өөрийнхөөрөө хэрэгжүүлдэг.
- *Зарлалт:* Эх классын гишүүн функцийн өмнө **virtual** гэх түлхүүр үгийг бичиж, их биеийг тодорхойлж өгснөөр тухайн функц нь хийсвэр функц болох юм.
- *Жишээ:* **class class_name{… virtual function_name(){…} …}**

```
class base {
   public:
   virtual void display() {
      cout<<"Function of base class"<<endl;
   }
};
```

- *Давуу тал:* Удамшлын үед класс хоорондын функцийн давхардлыг арилгах. Мөн эх классын гишүүн функцийг удамшуулахгүйгээр дахин програмчлах боломжийг олгодог.

3.3. Жинхэнэ хийсвэр функц гэж юу вэ? Тодорхойлолт, зарлалт, жишээ, давуу талыг тус тус бич.

- *Тодорхойлолт:* Жинхэнэ хийсвэр функц нь эх классийн гишүүн функц бөгөөд удамшуулах үед түүнийг хүү классууд дахин програмчилж ашиглах боломжтой функцийг хэлэх бөгөөд тухайн эх класс зарлагдсан жинхэнэ хийсвэр аргыг хүү классууд нь өөр өөрийнхөөрөө тодорхойлж хэрэгжүүлдэг.
- *Зарлалт:* Эх классын гишүүн функцийн өмнө **virtual** гэх түлхүүр үгийг бичиж, их биеийг тодорхойлохгүйгээр зарлаж өгснөөр тухайн функц нь жинхэнэ хийсвэр функц бол юм.
- *Жишээ:* **class class_name{… virtual function_name(){} = 0 …}**

```
class base {

   public:

   virtual void display() {}=0;

};
```

- *Давуу тал:* Удамшлын үед класс хоорондын функцийн давхардлыг арилгах. Мөн эх классын гишүүн функцийг удамшуулахгүйгээр дахин програмчлах боломжийг олгодог.

3.4. Функц дахин программчлах гэж юу вэ? Эх классын дахин программчилсан функцыг хүүхэд классын функц дотроос хэрхэн дууддаг вэ?

- Функц дахин программчлах гэдэг нь нэг класс зөвхөн өөр нэг классаас удамших үед тохиолддог бөгөөд эх класст зарлагдсан эсвэл зарлан тодорхойлсон функцийн биеийг дэд класст дахин бичих үйл явц юм.
- Эх классын дахин программчилсан функцыг хүүхэд классын функц дотроос шууд дуудах боломжтой.

3.5. Хийсвэр класс гэж юу вэ? Хэрхэн объект байгуулдаг вэ?

- Хийсвэр класс гэдэг нь тухайн классаас шууд обьект үүсгэн ашиглах боломжгүй, зөвхөн удамшуулан хэрэглэх, удамшил хоорондын давхардлыг арилгах зориулалттай класс юм.
- Хийсвэр классын объектыг өөрийг нь удамшуулж байж байгуулах боломжтой.

3.6. Удамшил ба устгагч функц хоёр ямар хамааралтай вэ?

- Удамших классын объект устах үед эхэнд удамших классын устгагч дараа нь эх классын устгагч функц дуудагдана.

4. **ХЭРЭГЖҮҮЛЭЛТ**

4.1. Lab07-д хийсэн дүрсүүдэд жинхэнэ хийсвэр функц нэмж

- Shape класст примтетер олох функцыг жинхэнэ хийсвэрээр зарлана.

```
class Shape{
protected:
	char* name;
	float side;
public:
	Shape(){
		name = new char(1);
		strcpy(name, "");
		float side = 1;
	}
	Shape(float a, char *name = ""){
		this->name = new char(strlen(name) + 1);
		strcpy(this->name, name);
		side = a;
	}
	~Shape(){
		delete name;
	}
	char* getName(){
		return name;
	}
	float getAside(){
		return side;
	}
	virtual float findPerimeter() = 0; // Периметр олох функцийг жинхэнэ хийсвэрээр // зарлав.
};
```

- TwoDimensionalShape класст талбай олох функцийг жинхэнэ хийсвэрээр зарлана.

```
class TwoDimensionalShape:public Shape{

protected:

	Point startVertex;

public:

	 TwoDimensionalShape(): Shape(){}

		 TwoDimensionalShape(Point vertex, float a, char*name = ""): Shape(a, name){

		startVertex = vertex;

	}

	virtual float findArea() = 0; // Талбай олох функцийг жинхэнэ хийсвэрээр зарлав.

};
```

4.2. Хийсвэр эх классын параметертэй, параметергүй байгуулагч функцийг хүүхэд класс бүрийн байгуулагч функцтай давхар дууддаг болгоно.

- Shape классаас удамшсан TwoDimensionalShape классын хувьд

```
class TwoDimensionalShape:public Shape{
protected:
        Point startVertex;
public:
        TwoDimensionalShape(): Shape(){}//Эх классын параметргүй байгуулагч давхар
        //дуудагдана.
        TwoDimensionalShape(Point vertex, float a, char*name = ""): Shape(a, name){
                //Эх классын параметртэй байгуулагч давхар дуудагдана.
                startVertex = vertex;
        }
        virtual float findArea() = 0;
};
```

- TwoDimensionalShape классаас удамшсан Circle, Square, Triangle классын хувьд

```
class Circle:public TwoDimensionalShape{
…
public:
        Circle():TwoDimensionalShape(){}//Эх классын параметргүй байгуулагч давхар //дуудагдана.
        Circle(Point ct, float r, char *ner = ""):TwoDimensionalShape(ct, r, ner){}//Эх //классын
        параметртэй байгуулагч давхар дуудагдана.
        …
}
```

```
class Square:public TwoDimensionalShape{
...
public:
        Square():TwoDimensionalShape(){//Эх классын параметргүй байгуулагч давхар //дуудагдана.
                Point A(-1, 1);
                startVertex = A;
                calcOtherPoints();
        }
        Square(Point lt_p, float a, char *name = ""):TwoDimensionalShape(lt_p, a, name){
        //Эх классын параметртэй байгуулагч давхар дуудагдана.
                setLTpoint(lt_p);
        }
        …
}
```

```
class RightTriangle:public TwoDimensionalShape{
…
public:
        RightTriangle(): TwoDimensionalShape(){//Эх классын параметргүй байгуулагч //давхар
        дуудагдана.
                Point A(0, 1);
                startVertex = A;
                calcOtherPoints();
        }
        RightTriangle(Point pt, float a, char *name = ""):TwoDimensionalShape(pt, a, name){
                //Эх классын параметртэй байгуулагч давхар дуудагдана.
                setTpoint(pt);
        }
        …
}
```

4.3. Харилцан адилгүй талбайтай олон ширхэг тойрог, гурвалжин, тэгш өнцөгтүүд үүсгэж хооронд нь талбайгаар нь эрэмбэл.

```
struct Shape_square{
   char name[10];
   float area;
};
typedef Shape_square Shape_square;

void add_shape_area(Shape_square* sh_a, char *n, float a){
   sh_a->area = a;
   strcpy(sh_a->name, n);
}

int main(){
        Shape_square sh_s[10];
        int shape_cnt = 0;
        Point ct(1, 1), … ;

        Circle c1(ct, 3, "C1");
        add_shape_area(sh_s[shape_cnt++], c1.getName(), c1.findArea());
        …
        //Insertion sort хэрэгжүүлэлт
        int i, j;
        float key;
        char n[10];
        for (i = 1; i < shape_cnt; i++)
        {
                key = sh_s[i].area;
                strcpy(n, sh_s[i].name);
                j = i - 1;

                while (j >= 0 && sh_s[j].area > key)
                {
                        sh_s[j + 1] = sh_s[j]; //Ямар нэг хаяг ашиглаагүй тул асуудал гарахгүй
                        j = j - 1;
                }
                sh_s[j + 1].area = key;
                strcpy(sh_s[j + 1].name, n);
        }
   return 0;
}
```

```
Circle:
CP(1, 1)
Name: C1
Radius: 3
Area: 28.2744
Perimeter: 18.8496

Square:
A(-2, 2)
B(0, 2)
C(0, 0)
D(-2, 0)
Name: S1
A side: 2
Area: 4
Perimeter: 8

RightTrangle:
A(2, 2)
B(3, 0.267949)
C(1, 0.267949)
Name: RT1
A side: 2
Area: 1.73205
Perimeter: 3
```

```
--Sorted shapes by area:

Name: RT1, Area: 1.732051

Name: S1, Area: 4.000000

Name: C1, Area: 28.274401
```

5. ДҮГНЭЛТ

Энэхүү лабораторын ажлаар жинхэнэ хийсвэр функц ашиглан лаборатор 7 – д хэрэгжүүлсэн классуудыг дахин загварчилсан. Ингэснээр эх классын функцийг удамшуулахгүйгээр дахин програмчилж байгаа тул эх болон дэд классын хооронд ямар нэг функцийн давхардалт үүсгэхгүй. Энэ нь аль классын функц дуудагдаж байна гэж эргэлзэхгүйгээр тухайн функцийг ашиглах боломжийг олгож буй явдал юм.

6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандлагат технологийн С++ програмчлал, Ж.Пүрэв, 2008, Улаанбаатар.
2. https://www.geeksforgeeks.org/function-overloading-vs-function-overriding-in-cpp/
3. https://techvidvan.com/tutorials/function-overloading-and-overriding-in-cpp/

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;
    float side;
public:
    Shape(){
        name = new char(1);
        strcpy(name, "");
        side = 1;
    }
    Shape(float a, char *name = ""){
        this->name = new char(strlen(name) + 1);
        strcpy(this->name, name);
        side = a;
    }
    ~Shape(){
        delete name;
    }
    char* getName(){
        return name;
    }
    float getAside(){
        return side;
    }
    virtual float findPerimeter() = 0;
};
class TwoDimensionalShape:public Shape{
protected:
    Point startVertex;
public:
    TwoDimensionalShape(): Shape(){}
    TwoDimensionalShape(Point vertex, float a, char*name = ""): Shape(a, name){
        startVertex = vertex;
    }
    virtual float findArea() = 0;
};

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;
```

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

```
169  struct Shape_square{
170      char name[10];
171      float area;
172  };
173  typedef Shape_square Shape_square;
174
175  void add_shape_area(Shape_square* sh_a, char *n, float a){
176      sh_a->area = a;
177      strcpy(sh_a->name, n);
178  }
179
180  int main(){
181      Point ct(1, 1), st(-2, 2), tt(2, 2);
182      Shape_square sh_s[10];
183      int shape_cnt = 0;
184
185      Circle c1(ct, 3, "C1");
186      cout<<"Circle: "<<endl;
187      c1.getData();
188      cout<<"Area: "<<c1.findArea()<<endl;
189      cout<<"Perimeter: "<<c1.findPerimeter();
190      cout<<"\n\n";
191      add_shape_area(&sh_s[shape_cnt++], c1.getName(), c1.findArea());
192
193      Square s1(st, 2, "S1");
194      cout<<"Square: "<<endl;
195      s1.getData();
196      cout<<"Area: "<<s1.findArea()<<endl;
197      cout<<"Perimeter: "<<s1.findPerimeter();
198      cout<<"\n\n";
199      add_shape_area(&sh_s[shape_cnt++], s1.getName(), s1.findArea());
200
201      RightTriangle rt1(tt, 2, "RT1");
202      cout<<"RightTrangle: "<<endl;
203      rt1.getData();
204      cout<<"Area: "<<rt1.findArea()<<endl;
205      cout<<"Perimeter: "<<rt1.findPerimeter();
206      cout<<"\n\n";
207      add_shape_area(&sh_s[shape_cnt++], rt1.getName(), rt1.findArea());
208
209      int i, j;
210      float key;
211      char n[10];
212      for (i = 1; i < shape_cnt; i++)
213      {
214          key = sh_s[i].area;
215          strcpy(n, sh_s[i].name);
216          j = i - 1;
217
218          while (j >= 0 && sh_s[j].area > key)
219          {
220              sh_s[j + 1] = sh_s[j];
221              j = j - 1;
222          }
223          sh_s[j + 1].area = key;
224          strcpy(sh_s[j + 1].name, n);
225      }
226
227      printf("\n\n--Sorted shapes by area: \n");
228      for(i = 0; i < shape_cnt; i++){
229          printf("\nName: %s, Area: %f\n", sh_s[i].name, sh_s[i].area);
230      }
231
232      return 0;
233  }
234
235
236
```