Хийсвэр Болон Жинхэнэ Хийсвэр Гишүүн Функцуудыг Тодорхойлох, Байгуулагч Функцын Ажиллагааг Судлах Лабораторын Ажил
(Лаборатори №09)

Н. Энхболд

ХШУИС, Програм хангамж 2-р түвшний оюутан, 20B1NUM0102

1. ОРШИЛ/УДИРТГАЛ

Амьдралд байгаа объектыг компьютерийн програмд загварчлахад C++ хэлийг ашиглан оюутан объектыг загварчилж оюутан бүртгэх, эрэмбэлэх зэрэг үйлдлийг олон объектод хийнэ.

2. ЗОРИЛГО

Долоодугаар лабораторийн ажлаар удамшлын талаарх ойлголтыг Shape классыг олон төрлөөр удамшуулах практикт тулгуурлан судалсан билээ. Энэхүү лаборатори дээр өмнөх лабораторийн ажлыг үргэлжлүүлэн судалж эдгээр зорилтуудыг тавьж ажиллав:

1. Хийсвэр функц гэж юу вэ? Тодорхойлолт, зарлал, жишээ, давуу талыг тус тус бичих
2. Жинхэнэ хийсвэр функц гэж юу вэ? Тодорхойлолт, зарлал, жишээ, давуу талыг тус тус бичих
3. Функц дахин программчлах гэж юу вэ? Эх классын дахин программчилсан функцыг хүүхэд классын функц дотроос хэрхэн дууддаг вэ?
4. Хийсвэр класс гэж юу вэ? Хэрхэн объект байгуулдаг вэ?
5. Удамшилд байгуулагч функц хэрхэн ашиглагддаг вэ?
6. Удамшил ба устгагч функц хоёр ямар хамааралтай вэ?

3. ОНОЛЫН СУДАЛГАА

3.1 Хийсвэр функц ба Жинхэнэ хийсвэр функц

Хийсвэр болон Жинхэнэ хийсвэр функцууд нь C++ програмчлалын хэл дээр virtual гэх түлхүүр үгийг ашиглан зарлагддаг бөгөөд тус тусын онцлог чанараараа ялгардаг.

Хийсвэр функц нь удамшил гэх ойлголтоор илрэх ба ямар нэгэн функцыг дахин програмчлах, давхардахаас сэргийлэх зорилготойгоор ашиглагддаг. Аливаа классын удамшсан буюу хүүхэд класст дахин өөрчилж болзошгүй функц тодорхойлохын тулд эх класст нь тухайн функцыг хийсвэрээр зарлаж өгдөг. Жишээ нь:

```
class Eh{
	public:
		virtual void print(){
			cout << "Eh class\n";
		}
};
class Hvvhed : public Eh{
	public:
		void print(){
			cout << "Hvvhed class\n";
		}
};

int main(){
	Eh *a;
	Hvvhed b;
	a = &b;
	a->print();
}
```

Дэлгэцэнд:

```
Hvvhed class
```

Жинхэнэ хийсвэр функц нь мөн адил удамшил гэх ойлголтоор илрэх ба хийсвэр функцээс ялгарах ялгаа нь тухайн жинхэнэ хийсвэр функцыг зарлаад орхидог буюу тодорхойлдоггүй ба агуулж буй класс нь хийсвэр класс болдог. Хийсвэр класс нь объект байгуулах чадавхигүй байдаг. Жишээлбэл:

```
class Eh{
	public:
		virtual void print() = 0;
};
class Hvvhed : public Eh{
```

```
	public:
		void print(){
			cout << "Hvvhed class\n";
		}
};

int main(){
	Hvvhed b;
	b.print();
}
```

Дэлгэцэнд:

```
Hvvhed class
```

3.2 Функц дахин програмчлах

Функц дахин программчлах гэдэг нь “Полиморфизм” гэж нэрлэгддэг ба хийсвэр функцтэй шууд холбогддог. Эх классын функцыг өөрчилж дахин ашиглахын тулд хийсвэр байдлаар зарлан тодорхойлж хүүхэд класстаа өөрийн хүссэн хэлбэрээр дахин тодорхойлж болдог. Хэрвээ жинхэнэ хийсвэр функц агуулсан классыг удамшуулан тухайн функцыг дахин програмчлахгүй үлдээвэл тухайн удамшсан класс нь эх класстайгаа адил хийсвэр класс болдог.

```
class Anhdagch{                         //Хийсвэр
	public:
		virtual void print() = 0;
};
class Hoyrdogch: public Anhdagch{       //Хийсвэр
};
class Guravdagch : public Hoyrdogch{    //Хийсвэр биш
	public:
		void print(){
			cout << "Polymorphism";
		}
};
```

3.3 Хийсвэр класс

Хийсвэр класс нь нэгэн төрлийн олон классуудын ерөнхий шинж чанарыг агуулсан объект үүсгэх чадавхигүй классыг хэлнэ. Гол зорилго нь удамшиж өөрөөсөө хүүхэд класс үүсгэх зорилготой бөгөөд тухайн хийсвэр классын объектыг үүсгэхийн тулд түүнийг удамшуулан тухайн хийсвэр класст буй жинхэнэ хийсвэр функцыг тодорхойлж өгөх хэрэгтэй. Хэрэв тодорхойлж өгөлгүйгээр шинэ класс удамшуулан бий болговол тухайн класс мөн адил жинхэнэ хийсвэр функц агуулж буй тул хийсвэр класс болно.

3.4 Байгуулагч функцын удамшилд буй ажиллагаа

Эх классын байгуулагч функц нь хүүхэд классын байгуулагч дуудагдахад түрүүлж ажилладаг бөгөөд хүүхэд класст буй байгуулагчаас эх классын байгуулагч руу хандахдаа “:” оператор ашигладаг. Жишээлбэл:

```
class Shape{
	protected:
		char *name;
	public:
		void setName(char *ner){
			name = new char[strlen(ner) + 1];
			strcpy(name, ner);
		}
		Shape(char *ner){
			name = new char[strlen(ner) + 1];
			strcpy(name, ner);
			cout << "******Dvrs vvssen******" << endl << endl;
		}
		Shape(){
			cout << "******Dvrs vvssen******" << endl << endl;
		}
}
class TwoDimensionalShape : public Shape{
	protected:
		int x;
		int y;
		int r;
	public:
		TwoDimensionalShape(char *ner, int a, int b, int urt) : Shape(ner){
```

```
                x = a;
                y = b;
                r = urt;
        }
};
```

Эх классын байгуулагч руу хүүхэд классын байгуулагч нь параметрээ дамжуулж өгөн эх классын байгуулагч ажилладаг.

3.5 Удамшил ба устгагч

Хэрэв эх класст устгагч функц тодорхойлсон бол хүүхэд класст онцын шаардлагагүй тохиолдолд тодорхойлохгүй байж болдог. Объектыг устгахад хүүхэд классын устгагч түрүүлж ажилладаг бөгөөд түүний араас эх классын устгагч ажилладаг.

4. ХЭРЭГЖҮҮЛЭЛТ

1. Лаборатори №07 дээр хийсэн лабораторийн ажлаа өргөтгөн жинхэнэ хийсвэр функцуудыг(findArea(), findPerimeter()) нэмж өгөн Shape болон TwoDimensionalShape классуудыг объект үүсгэх чадавхигүй болгов.

```
class Shape{
        protected:
                char *name;
        public:
                void setName(char *ner){
                        name = new char[strlen(ner) + 1];
                        strcpy(name, ner);
                }
                Shape(char *ner){
                        name = new char[strlen(ner) + 1];
                        strcpy(name, ner);
                        cout << "******Dvrs vvssen******" << endl << endl;
                }
                Shape(){
                        cout << "******Dvrs vvssen******" << endl << endl;
                }
                ~Shape(){
```

```
                delete name;
                cout << ">>Dvrs ustgagdaw<<" << endl;
        }
        virtual float findPerimeter() = 0;          //Жинхэнэ хийсвэр функц
};

class TwoDimensionalShape : public Shape{
    protected:
        int x;
        int y;
        int r;
    public:
        TwoDimensionalShape(char *ner, int a, int b, int urt) : Shape(ner){
            x = a;
            y = b;
            r = urt;
        }
        TwoDimensionalShape(char *ner) : Shape(ner){
        }
        int getRadius(){
            return r;
        }
        virtual float findArea() = 0;               //Жинхэнэ хийсвэр функц
};
```

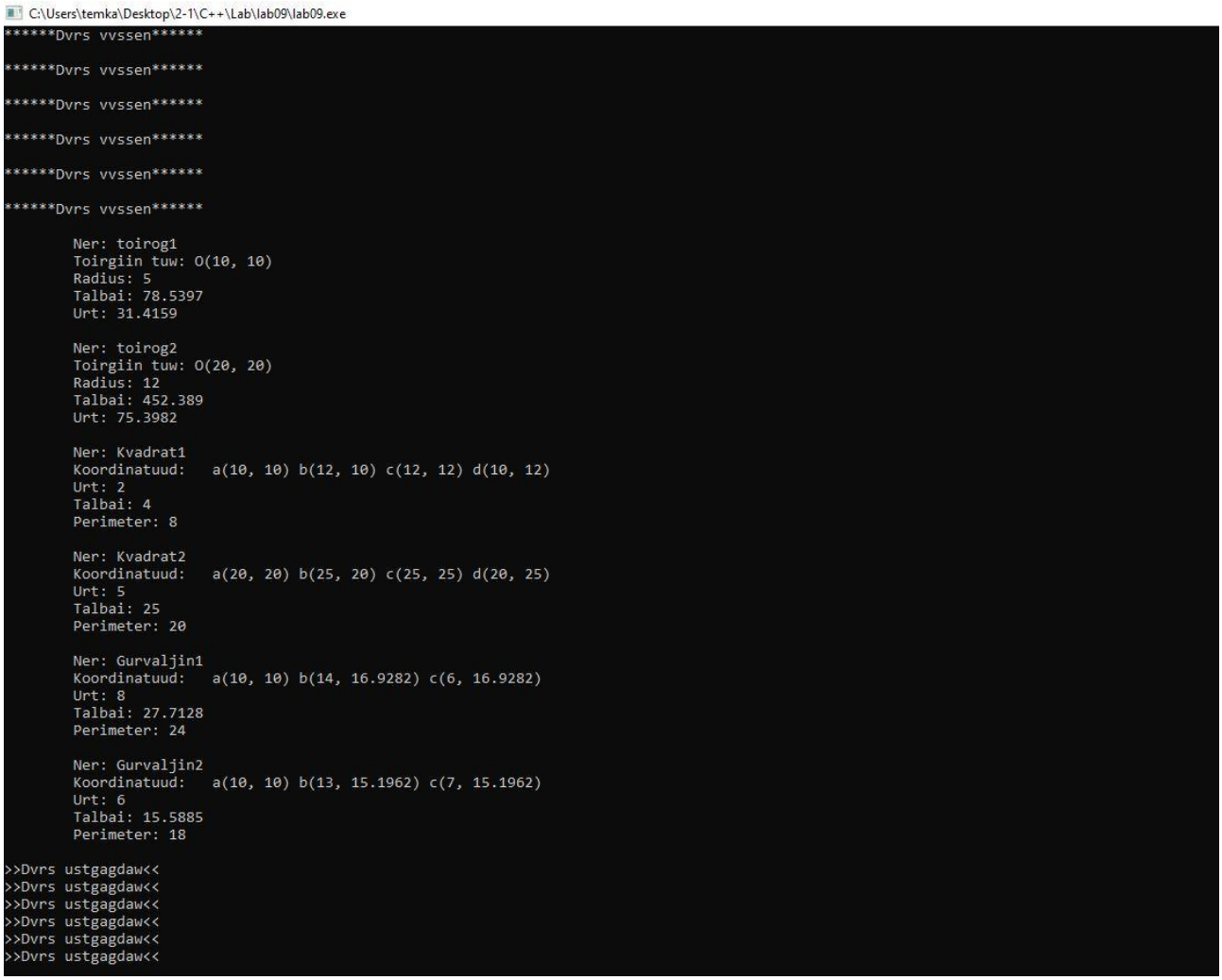
2. findArea(), findPerimeter() функцуудыг Circle, Square, Triangle гэх удамшсан классуудад тус бүр тохирох байдлаар нь тодорхойлж өгөв.

```
class Circle : public TwoDimensionalShape{
    public:
        float findArea(){
            return PI * r * r;
        }
        float findPerimeter(){
            return 2 * PI * r;
        }
```

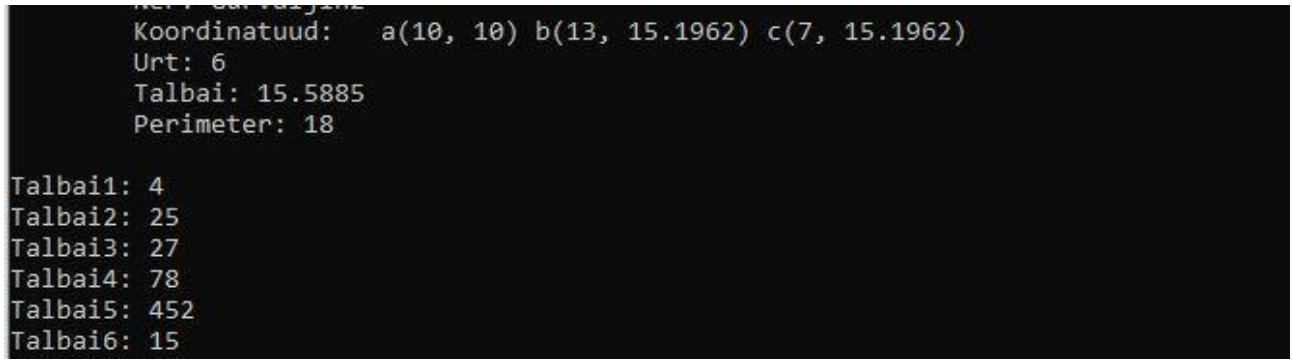
```
		Circle() : TwoDimensionalShape("Dugui"){
			x = 0;
			y = 0;
		}
		Circle(char *ner, int a, int b, int urt) : TwoDimensionalShape(ner, a, b,
urt){
		}
		void setRadius(float a){
			r = a;
		}
		void print(){
			cout <<"\tNer: " << name << endl;
			cout << "\tToirgiin tuw: O(" << x << ", " << y << ")" <<endl;
			cout << "\tRadius: " << r << " " << endl;
			cout << "\tTalbai: " << findArea() << endl;
			cout << "\tUrt: " << findPerimeter() << endl << endl;
		}
};

class Square : public TwoDimensionalShape{
	private:
		int x1, x2, x3, y1, y2, y3;
	public:
		float findArea(){
			return r * r;
		}
		float findPerimeter(){
			return 4 * r;
		}
		Square() : TwoDimensionalShape("Kvadrat"){
			x = 0;
			y = 0;
```

```
		r = 1;
		x1 = x + r;
		y1 = y;
		x2 = x + r;
		y2 = y + r;
		x3 = x;
		y3 = y + r;
	}
	Square(char *ner, int a, int b, int urt) : TwoDimensionalShape(ner, a, b,
urt){
		x1 = x + r;
		y1 = y;
		x2 = x + r;
		y2 = y + r;
		x3 = x;
		y3 = y + r;
	}
	void setLength(float l){
		r = l;
		x1 = x + r;
		y1 = y;
		x2 = x + r;
		y2 = y + r;
		x3 = x;
		y3 = y + r;
	}
	void setA(float a, float b){
		x = a;
		y = b;
		x1 = x + r;
		y1 = y;
		x2 = x + r;
```

```
		y2 = y + r;
		x3 = x;
		y3 = y + r;
	}
	void print(){
		cout <<"\tNer: " << name << endl;
		cout << "\tKoordinatuud: ";
		cout << "\ta(" << x << ", " << y << ") ";
		cout << "b(" << x1 << ", " << y1 << ") ";
		cout << "c(" << x2 << ", " << y2 << ") ";
		cout << "d(" << x3 << ", " << y3 << ") " << endl;
		cout << "\tUrt: " << r << " " << endl;
		cout << "\tTalbai: " << findArea() << endl;
		cout << "\tPerimeter: " << findPerimeter() << endl << endl;
	}
};
```



3. Circle, Square, Triangle классуудын объектууд хэд хэдийг үүсгэж талбайг areas[] массивт тэдгээрийн талбайг хадгалж статик байдлаар эрэмбэлэв.



5. ДҮГНЭЛТ

Хийсвэр функц, жинхэнэ хийсвэр функц болон хийсвэр класс гэх ойлголтуудыг Shape, TwoDimensionalShape гэх классуудад ашиглан объект үүсгэхээс сэргийлэв. Мөн жинхэнэ хийсвэр функцыг удамшуулан дахин тодорхойлж класс бүрээс хамааран өөрөөр ажиллах байдлаар тодорхойлж өгөв. Дахин програмчлалыг ашигласнаар тухайн хүүхэд класс бүрээс хамааран ажиллагаа нь өөрчлөгдөж болох функцууд дээр ашиглан функц давхардахаас сэргийлж, тухайн функцыг илүү үр дүнтэйгээр ашиглах боломжийг бий болгов.

6. АШИГЛАСАН МАТЕРИАЛ

1. Объект хандлагат програмчлал – Лекц №09 - Удамшил
https://drive.google.com/file/d/1_AhR0fH2RVLExpzUcGCOHrp4p0km1uyB/view?usp=sharing

2. Объект хандлагат програмчлал – Лекц №11 - Template, Virtual
https://drive.google.com/file/d/13KXsBdW_nCZ1XV-_-5k0kqnuXY9BROy7/view?usp=sharing

3. Объект хандлагат програмчлал – Лаборатори №07 – Удамшил

7. ХАВСРАЛТ

```
 #include <iostream>
#include <math.h>
#define PI 3.14159

using namespace std;
class Shape{
	protected:
		char *name;
	public:
		void setName(char *ner){
			name = new char[strlen(ner) + 1];
			strcpy(name, ner);
		}
		Shape(char *ner){
			name = new char[strlen(ner) + 1];
			strcpy(name, ner);
			cout << "******Dvrs vvssen******" << endl << endl;
		}
		Shape(){
```

```
            cout << "******Dvrs vvssen******" << endl << endl;
        }
        ~Shape(){
            delete name;
            cout << ">>Dvrs ustgagdaw<<" << endl;
        }
        virtual float findPerimeter() = 0;
};

class TwoDimensionalShape : public Shape{
    protected:
        int x;
        int y;
        int r;
    public:
        TwoDimensionalShape(char *ner, int a, int b, int urt) : Shape(ner){
            x = a;
            y = b;
            r = urt;
        }
        TwoDimensionalShape(char *ner) : Shape(ner){
        }
        int getRadius(){
            return r;
        }
        virtual float findArea() = 0;
};

class Circle : public TwoDimensionalShape{
    public:
        float findArea(){
            return PI * r * r;
        }
        float findPerimeter(){
```

```
                return 2 * PI * r;
        }
        Circle() : TwoDimensionalShape("Dugui"){
                x = 0;
                y = 0;
        }
        Circle(char *ner, int a, int b, int urt) : TwoDimensionalShape(ner,
a, b, urt){
        }
        void setRadius(float a){
                r = a;
        }
        void print(){
                cout <<"\tNer: " << name << endl;
                cout << "\tToirgiin tuw: O(" << x << ", " << y << ")" <<endl;
                cout << "\tRadius: " << r << " " << endl;
                cout << "\tTalbai: " << findArea() << endl;
                cout << "\tUrt: " << findPerimeter() << endl << endl;
        }
};

class Square : public TwoDimensionalShape{
    private:
        int x1, x2, x3, y1, y2, y3;
    public:
        float findArea(){
                return r * r;
        }
        float findPerimeter(){
                return 4 * r;
        }
        Square() : TwoDimensionalShape("Kvadrat"){
                x = 0;
                y = 0;
```

```
                r = 1;
                x1 = x + r;
                y1 = y;
                x2 = x + r;
                y2 = y + r;
                x3 = x;
                y3 = y + r;
        }
        Square(char *ner, int a, int b, int urt) : TwoDimensionalShape(ner,
a, b, urt){
                x1 = x + r;
                y1 = y;
                x2 = x + r;
                y2 = y + r;
                x3 = x;
                y3 = y + r;
        }
        void setLength(float l){
                r = l;
                x1 = x + r;
                y1 = y;
                x2 = x + r;
                y2 = y + r;
                x3 = x;
                y3 = y + r;
        }
        void setA(float a, float b){
                x = a;
                y = b;
                x1 = x + r;
                y1 = y;
                x2 = x + r;
                y2 = y + r;
                x3 = x;
```

```
                y3 = y + r;
        }
        void print(){
                cout <<"\tNer: " << name << endl;
                cout << "\tKoordinatuud: ";
                cout << "\ta(" << x << ", " << y << ") ";
                cout << "b(" << x1 << ", " << y1 << ") ";
                cout << "c(" << x2 << ", " << y2 << ") ";
                cout << "d(" << x3 << ", " << y3 << ") " << endl;
                cout << "\tUrt: " << r << " " << endl;
                cout << "\tTalbai: " << findArea() << endl;
                cout << "\tPerimeter: " << findPerimeter() << endl << endl;
        }
};

class Triangle : public TwoDimensionalShape{
      private:
            float x1, x2, y1, y2;
      public:
            float findArea(){
                  return (r * r * sqrt(3) /2 /2);
            }
            float findPerimeter(){
                  return 3 * r;
            }
            Triangle() : TwoDimensionalShape("Gurvaljin"){
            }
            Triangle(char *ner, int a, int b, int urt) :
TwoDimensionalShape(ner, a, b, urt){
                  y1 = r * cos(30 * PI /180.00) + y;
                  y2 = r * cos(30 * PI /180.00) + y;
                  x1 = r * sin(30 * PI/180.00) + x;
                  x2 = x - r * sin(30 * PI/180.00);
            }
```

```
        void setLength(float l){
            r = l;
            y1 = r * cos(30 * PI /180.00) + y;
            y2 = r * cos(30 * PI /180.00) + y;
            x1 = r * sin(30 * PI/180.00) + x;
            x2 = x - r * sin(30 * PI/180.00);
        }
        void setA(float a, float b){
            x = a;
            y = b;
            y1 = r * cos(30 * PI /180.00) + y;
            y2 = r * cos(30 * PI /180.00) + y;
            x1 = r * sin(30 * PI/180.00) + x;
            x2 = x - r * sin(30 * PI/180.00);
        }
        void print(){
            cout <<"\tNer: " << name << endl;
            cout << "\tKoordinatuud: ";
            cout << "\ta(" << x << ", " << y << ") ";
            cout << "b(" << x1 << ", " << y1 << ") ";
            cout << "c(" << x2 << ", " << y2 << ") " << endl;
            cout << "\tUrt: " << r << " " << endl;
            cout << "\tTalbai: " << findArea() << endl;
            cout << "\tPerimeter: " << findPerimeter() << endl << endl;
        }
};


int main(){
    int tmp, i, j;
    float areas[6];
    //Toirog
    Circle c1("toirog1", 10, 10, 5);
    Circle c2("toirog2", 20, 20, 12);
```

```
	//Kvadrat
	Square s1("Kvadrat1", 10, 10, 2);
	Square s2("Kvadrat2", 20, 20, 5);

	//Gurvaljin
	Triangle t1("Gurvaljin1", 10, 10, 8);
	Triangle t2("Gurvaljin2", 10, 10, 6);
	c1.print();
	c2.print();
	s1.print();
	s2.print();
	t1.print();
	t2.print();

	areas[0] = c1.findArea();
	areas[1] = c2.findArea();
	areas[2] = s1.findArea();
	areas[3] = s2.findArea();
	areas[4] = t1.findArea();
	areas[5] = t2.findArea();
	for(i = 0; i < 5; i++){
	   for(j = i + 1; j < 5; j++){
	      if(areas[j] < areas[i]){
	         tmp = areas[i];
	         areas[i] = areas[j];
	         areas[j] = tmp;
	      }
	   }
	}
	for(i = 0; i < 6; i++) {
	      cout << "Talbai" << i+1 << ": " << areas[i]<<"\n";
	}
}
```