

# (Лаборатори №9)

Э.Багабанди

ХШУИС - ийн ПХ хөтөлбөрийн

3-р түвшний оюутан, [19B1NUM0700@num.edu.mn](mailto: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 объект 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
```

```
RightTriangle:
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 C++ 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