C++ хэлний байгуулагч, устгагч функц болон хандалтын түвшин, функцийг дахин тодорхойлох Лабораторі 4

Э.Түвшин-Эрдэнэ(23B1NUM0869)
МТЭС, МКУТ-ын “Программ хангамж” хөтөлбөрийн оюутан, tuvshin.erdene25@gmail.com
2025 он

1.Оршил

Энэ лабораторийн ажлаар байгуулагч ба устгагч функц, хандалтын түвшин, функцийг дахин тодорхойлох аргачлал зэргийг судлан, өмнөх лабораторийн ажлаар зарласан класст тохирох өөрчлөлтүүдийг хийн кодыг “Хэрэгжүүлэлт”, “хавсралт” хэсэгт оруулав.

2.Зорилго

C++ хэл дээр байгуулагч болон устгагч функцийг хэрхэн ашиглах, өөр өөр хандалтын түвшингүүдийн ялгаа, функцийг дахин тодорхойлох талаар судлан C++ дээр жишээ бодлогоор туршин үзэх.

3. Онолын судалгаа

3.1)Байгуулагч функц, анхдагч байгуулагч функц

Нэр нь классын нэртэй ижил байх гишүүн функцийг класс дотор нэмж тодорхойлон объектод ой бэлдэх, гарааны утга оноох ажлыг түүнд хариуцуулж болдог. Ийм функцийг **constructor** буюу байгуулагч функц, **байгуулагч** гэж нэрлэдэг.[1]

Байгуулагч функц нь утга буцаахгүй ч гэсэн void гэж зааж өгдөггүй. Мөн байгуулагч функцийг дахин тодорхойлж болдог.

Байгуулагч тодорхойлох загвар:

```
<class name>(<parameters>){

        //constructor body
}
```

Хэрэв программ хөгжүүлэгч ямах нэгэн байгуулагч тодорхойлж өгөөгүй байвал C++ компайлер байгуулагч функцийг программд нэмж тодорхойлдог. Ийм байгуулагчийг **default constructor** буюу **анхдагч байгуулагч** гэж нэрлэдэг.
Гэхдээ анхдагч байгуулагчийг илээр тодорхойлж болох ба ийм байгуулагч гарааны утга бүхий параметртэй эсвэл параметргүй байгуулагч байна.

3.2) Устгагч функц

Цаашдаа хэрэглэхгүй объектыг устгахдаа түүний эзэмшиж байсан санах ойг чөлөөлөх зорилготой функцийг **устгагч функц(destructor)** гэх ба устгагч функцгүйгээр объект устаж чадахгүй.

Хэрэв класс дотор устгагч функцийг тодорхойлж өгөөгүй бол C++ компайлер анхдагч устгагч функцийг нэмж тодорхойлж хэрэглэнэ.

Устгагч функцийг тодорхойлох:

```
~<class name>(){
        //function body
}
```

Устгагч функц нь дараах онцлогтой:

- Утга буцаахгүй ба параметр авахгүй.
- Хаягаар нь хандаж болохгүй.
- Классын бусад функцийн адилаар удамшдаггүй.
- Класс ганц устгагч функцтэй байна.
- Дотоод мөр функц байж болно.

3.3) Функц дахин тодорхойлох(function overriding)

Ижил нэртэй боловч параметрийн тоо, төрөл өөр байх функцуудыг зарлан тодорхойлох ба C++ хэлний компайлер дуудагдаж буй функцийн аргументийн тоо, төрлийг функцийн тодорхойлолтын толгой хэсгийн бичдэстэй тулган шалгах замаар аль функцийг дуудахаа шийддэг.

Мөн ижил нэртэй функцуудын буцаах утгын төрөл ижил байж болно.
Жишээ:

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

class Student{
    int i;
    float f;

    public:
        Student(): i(2), f(5.5){}
        float square(float f){
            this->f = f*f;
            return this->f;
        }
```

```
        int square(int i){
            this->i=i*i;
            return this->i;
        }
        void show(){
            cout<<"f= "<<f<<"\ni= "<<i;
        }
};

int main(){
    Student aa;
    aa.show();
    float ff = aa.square((float)3.5);
    int ii = aa.square(3);
    cout <<endl<<endl<<"ff= "<<ff<<"\nii= "<<ii<<endl;

    return 0;
}
```

```
f= 5.5
i= 2

ff= 12.25
ii= 9
```

3.4) Хандалтын түвшин

Класс нь түүний гишүүд рүү хандах хандалтын private, public, protected гэсэн гурван түвшинтэй ба дараах ялгаатай:

- Private- Зөвхөн тухайн класс дотроос хандаж болно.
- Public - Классын гаднаас өөр класс эсвэл функцээс хандаж болно.
- Protected - Private-тэй төстэй гэхдээ удамшсан классын объектуудаас хандаж болно.

Классын тодорхойлолт дотор тусгай үгийн үйлчлэх хүрээ нь дараагийн тусгай үг эсвэл тодорхойлолтын төгсгөлийн хаах их хаалт гарах хүртэл үргэлжилнэ.

Бүтцийн гишүүд анхнаасаа public шинжтэй байх ба классын хувьд хэрэв тусгайлан зааж өгөөгүй бол private байна. Иймд private түлхүүр үгийг ашиглахгүй байж болно.

```
Class employee{
        Char name[20];
        Int basicpay;
        Int allowance;
        Public:
        Void getdata(void);
        Void showdata(void);
};
```

3.5) Динамик санах ой болон байгуулагч, устгагч функц

Шинэ объектод ой бэлдэхдээ “new” оператор хэрэглэж болох ба уг оператор объектод бэлдсэн ойн эхлэл хаягийг буцаана.

```
Class_name *pointer_name = new  class_name
```
Энэ жишээнд гарааны утга мөн оноож болно.

Хаяган хувьсагчаар дамжуулан классын гишүүнд хандахдаа сум эсвэл шууд хандалтын цэгийг ашиглаж болно:
```
pointer->(attribute||method);
(*pointer).(attribute||method);
```

Үүний адилаар “delete” операторoор ойг чөлөөлж болно.

```
Delete pointer_name
```

4. Хэрэгжүүлэлт

4.1) Ажилтны класст байгуулагч болон устгагч функцуудыг тодорхойлов:

```
Employee(int id = 0, char name[] = "", char position[] = "worker", float
worked_hours = 0, float hourly_salary = 0)
    {
        this->id = id;
        strcpy(this->name, name);
        strcpy(this->position, position);
        this->worked_hours = worked_hours;
        this->hourly_salary = hourly_salary;
    }
    ~Employee()
```

```
    {
        std::cout << "\n the object with " << id << " id  is
destroyed.\n";
    }
```

Устгагч функц нь объектын үйлчлэх хүрээнээс гарахад дуудагдаж байна.

4.2) Ажилтны классын захирлын цалин бодох функц болон бусад гишүүн өгөгдлүүдийг private болгон өөрчлөв.

```
class Employee
{

    int id;
    char name[20];
    char position[10];
    float worked_hours;
    float hourly_salary;
    float bossSalary()
    {
        // boss iin bonus iig bodoj olno.
        return worked_hours * hourly_salary * 1.5;
    }
```

Энд public хандалтын түвшнээс өмнө тодорхойлсон тул private байна.

4.3) getter setter функцуудыг тодорхойлов:

```
int getID()
    {
        return this->id;
    }
    char *getName()
    {
        return this->name;
    }
    char *getPosition()
    {
        return this->position;
    }
    float getWorked_hours()
```

```
    {
        return this->worked_hours;
    }
    float getHourly_salary()
    {
        return this->hourly_salary;
    }

    void setID(int id)
    {
        this->id = id;
    }
    void setName(char name[])
    {
        strcpy(this->name, name);
    }
    void setPosition(char position[])
    {
        strcpy(this->position, position);
    }
    void setWorked_hours(float worked_hours)
    {
        this->worked_hours = worked_hours;
    }
    void setHourly_salary(float hourly_salary)
    {
        this->hourly_salary = hourly_salary;
    }
```

4.4) Ажилчны классын динамик хүснэгт үүсгэв:

```
vector<Employee> v;
```

4.5) Insertion sort ашиглан цалингаар эрэмбэлэх боломжтой болгов:

```
for (int i = 0; i < v.size(); i++)
            {
                int j = i;
                while (j > 0 && v[j].calcSalary()>v[j-1].calcSalary())
                {
```

```
                    Employee temp = v[j];
                    v[j] = v[j - 1];
                    v[j - 1] = temp;
                    j--;
                }
            }
            cout<<"Sorted!!!"<<endl;
```

5.Дүгнэлт

Байгуулагч функц нь объектыг үүсгэх үед гарааны утга оноох үйлдлийг илүү оновчтой болгож байгаа ба устгагч функцийг тодорхойлсноор объект программын аль хэсэгт устаж байгааг мэдэх боломжтой болж байна. Мөн функцийг дахин тодорхойлсноор ижил нэртэй функцууд параметрээс шалтгаалан өөр өөр үйлдэл хийж болж байгаа нь давуу тал болж байна. Мөн хандалтын түвшинг зааж өгснөөр өгөгдлийн бүрэн бүтэн байдал, нууцлалд эерэгээр нөлөөлж байгаа ба стандарт өгөгдлийн төрлийн адил объектын хаяган хувьсагчтай new болон delete операторуудыг ашиглаж болж байна.

Объектын хаяган хувьсагчийг үргэлжлүүлэн судлах шаардлагатай санагдав.

6.Ашигласан материал

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

7.Хавсралт

Lab04.cpp

```
#include "Employee.h"
using namespace std;

int main()
{
    cout << "Human resource program by Tuvshin-Erdene 2025" << endl;
    vector<Employee> v;

    while (1)
    {
        int choice;
```

```
        cout << "1. list of employees 2. choose employee 3.new employee
4.sort by salary -1 to exit: ";
        cin >> choice;

        if (choice == 1)
        {
            for (int i = 0; i < v.size(); i++)
            {
                cout << i << "." << endl;
                v[i].getInfo();
            }
            cout << "------------" << endl;
            cout << "total number of employees: " << v.size() << endl;
        }

        else if (choice == 2)
        {
            cout << "Enter the id: ";
            int cID, num;
            cin >> cID;
            for (int i = 0; i < v.size(); i++)
            {
                if (v[i].getID() == cID)
                {
                    num = i;
                    break;
                }
            }
            v[num].getInfo();
            cout << endl;
            int choice2;
            while (1)
            {
                cout << "1.set information 2.calculate salary 3.increase
work hour -1 to exit: ";
                cin >> choice2;
                if (choice2 == 1)
                {
                    v[num].setInfo();
                }
```

```
            else if (choice2 == 2)
            {
                float salary = v[num].calcSalary();
                cout << "salary = " << salary << endl;
            }
            else if (choice2 == 3)
            {
                cout << "enter hour: ";
                int time;
                cin >> time;
                v[num].inc_hour(time);
            }

            else
            {
                cout << "exiting employee with id : " << cID << endl;
                break;
            }
        }
    }
    else if (choice == 3)
    {
        Employee e1;
        e1.setInfo();
        v.push_back(e1);
        cout << "Added successfully" << endl;
    }
    else if (choice == 4)
    {

        for (int i = 0; i < v.size(); i++)
        {
            int j = i;
            while (j > 0 && v[j].calcSalary()>v[j-1].calcSalary())
            {
                Employee temp = v[j];
                v[j] = v[j - 1];
                v[j - 1] = temp;
                j--;
            }
```

```
            }
            cout<<"Sorted!!!"<<endl;
        }
        else
        {
            break;
        }
    }
    return 0;
}
```

Employee.h

```
#include <iostream>
#include <vector>
#include <cstring>


class Employee
{

    int id;
    char name[20];
    char position[10];
    float worked_hours;
    float hourly_salary;
    float bossSalary()
    {
        // boss iin bonus iig bodoj olno.
        return worked_hours * hourly_salary * 1.5;
    }

public:
    Employee(int id = 0, char name[] = "", char position[] = "worker",
float worked_hours = 0, float hourly_salary = 0)
    {
        this->id = id;
        strcpy(this->name, name);
        strcpy(this->position, position);
        this->worked_hours = worked_hours;
        this->hourly_salary = hourly_salary;
```

```
    }
    ~Employee()
    {
        std::cout << "\n the object with " << id << " id  is
destroyed.\n";
    }

    void setInfo()
    {

        std::cout << "enter id: ";
        std::cin >> id;
        std::cout << "enter name: ";
        std::cin >> name;
        std::cout << "enter position: ";
        std::cin >> position;
        std::cout << "enter worked hours: ";
        std::cin >> worked_hours;
        std::cout << "enter hourly salary: ";
        std::cin >> hourly_salary;
    }
    void getInfo()
    {
        std::cout << "ID: " << id << std::endl;
        std::cout << "name: " << name << std::endl;
        std::cout << "position: " << position << std::endl;
        std::cout << "worked hours: " << worked_hours << std::endl;
        std::cout << "hourly salary: "<<hourly_salary<<std::endl;
    }

    float calcSalary()
    {
        if (strcmp(position, "boss") == 0)
        {
            return bossSalary(); // herew boss bol bonus nemj bodno.
        }
        return hourly_salary * worked_hours;
    }

    bool inc_hour(float time)
```

```
{
    if (time < 0 || time > 24)
    {
        return 0;
    }
    worked_hours += time;
    return 1;
}

int getID()
{
    return this->id;
}
char *getName()
{
    return this->name;
}
char *getPosition()
{
    return this->position;
}
float getWorked_hours()
{
    return this->worked_hours;
}
float getHourly_salary()
{
    return this->hourly_salary;
}

void setID(int id)
{
    this->id = id;
}
void setName(char name[])
{
    strcpy(this->name, name);
}
void setPosition(char position[])
{
```

```
        strcpy(this->position, position);
    }
    void setWorked_hours(float worked_hours)
    {
        this->worked_hours = worked_hours;
    }
    void setHourly_salary(float hourly_salary)
    {
        this->hourly_salary = hourly_salary;
    }
};
```