Байгуулагч Функц Болон Устгагч Функцийн Ажиллагааг Судлах, Хуулагч Байгуулагчийг Тодорхойлох Лабораторийн Ажил (Лаборатори №5)

Н. Энхболд

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

1. ОРШИЛ/УДИРТГАЛ

Байгуулагч болон устгагч функцуудыг хэзээ ажилдгийг судлан түүн дээрээ динамик санах ойг илүү бага зардлаар бий болгон чөлөөлөв. Мөн санах ойн цоорхойг үүсч болох нөхцөлүүдийг тооцон хуулагч байгуулагчийг тодорхойлж эрэмбэлэх алгоритм дээрээ объектуудын байрыг сэлгэхэд ашигласан бөгөөд объектон хүснэгтийн давуу талуудыг sorting болон ‘->’ хандалтын операторын тусламжтайгаар энэхүү лабораторийн ажил дээр харуулж өгөв.

2. ЗОРИЛГО

Өмнөх лабораторийн ажил дээр хийсэн ажилчин класс дээр судалгаа хийж:

- Байгуулагч, устгагч функцууд хэзээ хэрхэн ажиллаж байгааг тодорхойлох

- Объектын хаяган хувьсагчийг ашиглан олон ажилчны мэдээллийг оруулж хаягаар нь эрэмбэлэх

- Эрэмбэлэхдээ хуулагч байгуулагчийн тусламжтайгаар объектод объектын утгуудыг оноож өгөх

- Байгуулагч функцд new операторыг ашиглан динамик санах ойг аль болох бага зардлаар үүсгэх зэргийг зорилгоо болгов.

3. ОНОЛЫН СУДАЛГАА

3.1 Байгуулагч функцын ажиллагаа

C++ хэл нь кодыг дээрээс доош мөр бүрээр уншдаг бөгөөд тухайн мөр дотроо зүүнээс баруун руу гэх дарааллаар кодыг ажиллуулдаг. Байгуулагч функцийн ажиллагаа нь үүний адилаар main() функц дотор объектыг зарлавал тухайн main() функцын дээрээс доош ажиллах байдлаар тухайн объектыг зарлах мөрөн дээр очих үед байгуулагч функц дуудагддаг.

```
class Test
{
      public:
      Test();
};
Test::Test()  {
    cout << "Constructed\n";
```

```
}
int main() {
    cout << "main() started\n";
    Test obj1;
    Cout << "main() end";
    return 0;
}

/*Дэлгэцэнд:   main() started
               Constructed
              main() end
```

3.2 Устгагч функцын ажиллагаа

C++ compiler нь тухайн объектыг устгагч функц дуудаж устгаагүй тохиолдолд програм ажиллаад дуусах үед хэрвээ устгагч функцийг тодорхойлсон бол тэрхүү функцыг дуудаж устгадаг.Эс бөгөөд compiler өөрөө устгагч функцийг тодорхойлон объектуудыг устгаж өгдөг.

Мөн бид устгагч функцийг өөрөө дуудаж устгах боломжтой байдаг.

```
class test{

public:
    int a;
    test(int m)
    {
        a = m;
    }

    ~test(){
    cout << "Deleted" << day;
    }
};


int main()
{
  test obj1(2);
  obj1.~test();
  cout << obj1.test;
  return 0;
}
/*Дэлгэцэнд:   Deleted
```

3.3 Хуулагч байгуулагч

Хуулагч байгуулагч нь parameter дээрээ тухайн классыг объектыг авдаг бөгөөд тэрхүү авсан parameter – ийнхээ бүх гишүүдийн утгыг тухайн функц дотроо ашиглаж болдог. Хуулагч функцийг дахин тодорхойлохын давуу талаас дурьдвал:

Бид ямар нэг тоо эсвэл тэмдэгт зэргийг утга оноох байдлаар шууд өгч чаддаг. Гэвч тухайн объект хэрэв өөртөө хаяган хувьсагч агуулсан байвал бид тухайн хаягийг шууд оноож өгч болохгүй билээ. Үүнд санах ойн цоорхой бий болох аюултай тул бид хуулагч функцийг дахин тодорхойлж тухайн тухайн хаягийг шууд оноож буй хэсгийг өөрчилж болох билээ.

Үүний тусламжтайгаар бид 1 объектын утгыг нөгөөд эрсдэл багатайгаар оноон өгч болох юм.

```
class Employee{
     public:
     int id;
     char *name;
     char *position;
     float workHour;
     Employee(int dugaar, char *ner, char *tushaal, float tsag){
               name = new char(strlen(ner));
               position = new char(strlen(tushaal));
               id = dugaar;
               strcpy(name, ner);
               strcpy(position, tushaal);
               workHour = tsag;
          }
     Employee(Employee &X){
               id = X.id;
               strcpy(name, X.name);
               strcpy(position, X.position);
               workHour = X.workHour;
          }
}
int main(){
     Employee a(1, ”Bat”, ”Manager”,15);
     Employee b(a);
}
```

Энэ тохиолдолд a объектын утгуудыг b-д санах ойн цоорхой үүсгэлгүйгээр шууд оноох боломжтой болж байгаа билээ.

3.4 Хуулагч функц

Дан хуулагч функц нь хуулагч байгуулагчаас parameter дээрээ объект биш гишүүдийн утгыг авдгаараа ялгаатай. copyFunction гэх гишүүн функц дээр жишээ авч үзье.

```
void copyFunction(int dugaar, char *ner, char *tushaal, float
tsag){
      name = new char(strlen(ner));
      position = new char(strlen(tushaal));
      id = dugaar;
      strcpy(name, ner);
      strcpy(position, tushaal);
      workHour = tsag;
}
```

Тухайн функцд дамжуулж өгч буй утгуудаа тухайн объектын гишүүдэд оноодог бөгөөд Параметрт байгуулагчийн адилаар Динамик санах ой бий болгож өгч болно. Үүнийгээ устгагч функц дээр delete оператор ашиглан устгаж өгөх байдлаар санах ойн цоорхойгоос сэргийлэх боломжтой билээ.

3.5 Объектын хаяган хувьсагч

Нэгэн төрлийн олон объект байгуулах гэж байгаа бол тухайн объектын хаяган хувьсагчийг зарлаж түүндээ тааруулан санах ой нөөцөлж болдог билээ. Ингэснээр объект бүрд нэр өгөн тус тусад нь хандах шаардлагагүй болно. Жишээ кодоор харуулбал:

```
int main(){
      Employee *ajilchid;                    //Хаяган хувьсагч зарлах мөр
      ajilchid = new Employee[5]             //5 объект бүхий санах ой тухайн хаяганд нөөцлөх
      /*Хэрвээ Employee классын ‘id’ гишүүн өгөгдлийг public горимтой гэж үзэн хандвал*/
      ajilchid [2]->id = 2;    //3-р ажилчин объектын id-д 2 гэх утга оноож буй үйлдэл юм.
}
```

Объектын гишүүдэд хандахдаа ‘.’ Оператор ашигладаг бол

Объектын хүснэгтийн гишүүдэд хандахдаа ‘->’ операторыг ашигладаг.

4. ХЭРЭГЖҮҮЛЭЛТ

1. Байгуулагч функцуудээ дахин тодорхойлсон бөгөөд Параметрт байгуулагч дээр name болон position гишүүдэд орж ирж буй тэмдэгтэн мөрүүдийн утгыг strlen оператор ашиглан тухайн урттай тэнцүү хэмжээний динамик санах ойг name болон position гишүүдэд нөөцөлж өгөв.

Тухайн нөөцөлсөн санах ойг санах ойн цоорхойгоос сэргийлэх байдлаар устгагч функц дээрээ delete оператор ашиглан name, position гишүүдийн санах ойг чөлөөлдөг байдлаар классыг тодорхойлсон.

```
Employee(int tsagiinHuls, int dugaar, char *ner, char *tushaal, float
tsag){
                name = new char(strlen(ner));
                position = new char(strlen(tushaal));
                id = dugaar;
                strcpy(name, ner);
                strcpy(position, tushaal);
                workHour = tsag;
                m++;
          }
~Employee(){
                delete name;
                delete position;
                cout << "Deleted successfully" << m <<endl;
                m--;
          }
```

2. Олон ажилчдыг объектон хүснэгтийн тусламжтайгаар бүртгэж тус бүрийн input() аргыг дуудаж гишүүн өгөгдлүүдэд нь утгыг оноосон бөгөөд эдгээрийн тус бүрийн цалинг бодон багаас нь их рүү байдлаар эрэмбэлж тухайн нэр болон нийт цалингийн хэмжээ харгалзан байхаар объектын хаягаар дамжуулан sort хийв.

```
for(i = 0; i < 3; i++){
          for(j = i + 1; j < 3; j++){
                if(total[j] < total[i]){
                      tmp = ajilchid[i];
                      ajilchid[i] = ajilchid[j];
                      ajilchid[j] = tmp;
```

```
                temp = total[i];
                total[i] = total[j];
                total[j] = temp;
                }
          }
     }
```

5. ДҮГНЭЛТ

Динамик санах ойг strlen оператортой хавсарган ашигласнаар илүү бага зардлаар санах ойн цоорхойгоос сэргийлэх давуу талтай бөгөөд destructor функц дээрээ уг санах ойг чөлөөлөх нь илүү хялбар шийдэл байв. Объектын хаяган хувьсагчийн тусламжтайгаар объект бүрд нэр өгөх шаардлагагүйгээр тухайн нэгэн төрлийн объектуудыг давталт ашиглан sort, input зэргийг хийх боломжтой болсон бөгөөд sort дотроо объектуудын байрыг солихын тулд хуулагч байгуулагчийг дахин тодорхойлсон ба ингэснээр объектынхоо гишүүн хаяган хувьсагчдын санах ойн цоорхойгоос сэргийлэх боломжийг олгов.

6. АШИГЛАСАН МАТЕРИАЛ

1. Хуулагч байгуулагч – Объект хандлагат програмчлал – Лекц 05.1
https://drive.google.com/file/d/1XnHj4bpb-x0CNJW1eSugWtKwCEc6A19N/view?usp=sharing

2. Байгуулагч болон устгагч - Объект хандлагат програмчлал – Лекц 05.2
https://drive.google.com/file/d/1JCbme_zvh-7P-7KJ_KfSKw4lAnR2DRuZ/view?usp=sharing

7. ХАВСРАЛТ

```
#include <iostream>
using namespace std;
/*Ажилчны Классийг Тодорхойлох*/
class Employee{
     private:
          static int m = 0;
          int id;
          char *name;
          char *position;
          float workHour;
          int hourSalary;
          float ceoSalary();
```

```
public:
        void input();
        void output();
        float salary();
        bool addHour(float );
        void printName(){
                cout << name << "\t";
        }
        Employee(){
                name = new char[20];
                position = new char[10];
                m++;
        }
        Employee(int tsagiinHuls){
                hourSalary = tsagiinHuls;
                m++;
        }
        Employee(int tsagiinHuls, int dugaar, char *ner, char
*tushaal, float tsag){
                name = new char(strlen(ner));
                position = new char(strlen(tushaal));
                id = dugaar;
                strcpy(name, ner);
                strcpy(position, tushaal);
                workHour = tsag;
                m++;
        }
        Employee(Employee &X){
                id = X.id;
                strcpy(name, X.name);
                strcpy(position, X.position);
                workHour = X.workHour;
        }
        ~Employee(){
                delete name;
```

```
                delete position;
                cout << "Deleted successfully" << m <<endl;
                m--;
            }

};
/*Гараас утга авах функц*/
void Employee::input(){
      cout << "ID: ";
      cin >> id;
      cout << "Ner: ";
      cin >> name;
      cout << "Tushaal: ";
      cin >> position;
      cout << "Ajillasan tsag: ";
      cin >> workHour;
      cout << "Tsagiin tsalin: ";
      cin >> hourSalary;
}
/*Мэдээлэл дэлгэцлэх*/
void Employee::output(){
      cout << "ID: " << id << endl;
      cout << "Ner: " << name << endl;
      cout << "Tushaal: "<< position << endl;
      cout << "Ajillasan tsag: " << workHour << endl;
}
/*Нийт цалин бодох*/
float Employee::salary(){
      int total = 0, add;
      cout << "Nemelt tsag: ";
      cin >> add;
      if(addHour(add))
            workHour += add;
      else
```

```
            cout << "Buruu utga oruulsan\n";

      if(strcmp(position, "ceo") == 0)
            total = ceoSalary() + hourSalary*workHour;
      else
            total =  hourSalary*workHour;
      return total;
}
/*Захирлын цалин бодох*/
float Employee::ceoSalary(){
      int hour = 20000;
      return hour*workHour;
}
/*Ажилласан цаг нэмэгдүүлэх*/
bool Employee::addHour(float nemelt){
      if(nemelt > 0 && nemelt <= 24){
            return 1;
      }
      return 0;
}

int main(){
      int total[3];
      Employee ajilchid[3], tmp;
      for(int i = 0; i < 3; i++){
            (ajilchid+i)->input();
            *(total+i) = (ajilchid+i)->salary();
            cout << endl << "Niit tsalin: " << *(total+i) << endl;
      }

      int temp, j, i;
      for(i = 0; i < 3; i++){
            for(j = i + 1; j < 3; j++){
                  if(total[j] < total[i]){
```

```
                    tmp = ajilchid[i];
                    ajilchid[i] = ajilchid[j];
                    ajilchid[j] = tmp;
                temp = total[i];
                total[i] = total[j];
                total[j] = temp;
                }
            }
        }
        cout << "Tsalingiin huvaarilalt"<< endl;
        for(i = 0; i < 3; i++){
            ajilchid[i].printName();
            cout << total[i]<< endl;
        }
}
```