(Лаборатори №4)

Э.Багабанди

ХШУИС - ийн ПХ хөтөлбөрийн

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

1. **ОРШИЛ/УДИРТГАЛ**

Уг лабораторийн хүрээнд өгөгдсөн асуултуудад хариулж, тайлбарлахад ашиглагдсан C++ хэлний онолын ойлголт(байгуулагч функц, устгагч функць функц дахин тодорхойлох, динамик санах ой) - ыг судалж, үндсэн олйголтуудыг эзэмшиж, лабораторт хэрэгжүүлсэн болно.

2. **ЗОРИЛГО**

Энэхүү лабораторийн ажлын хүрээнд C++ хэлний байгуулагч функц, устгагч функць функц дахин тодорхойлох, динамик санах ой (new, delete оператор ашигласан) болон байгуулагч, устагч функцийг хэрхэн хамтад нь ашиглах онолын ойлголтуудыг судалж, үүнийг ашиглан лабораторт өгөгдсөн бодлого болон асуултуудыг шийдвэрлэж, мөн жижиг хэмжээний програм хөгжүүлэн гарч буй үр дүнг тайлбарлана.

3. **ОНОЛЫН СУДАЛГАА**

3.1 Байгуулагч ба устгагч функц

- **Байгуулагч функц**
  - Байгуулагч функц нь класстайгаа ижил нэртэй байна. Гишүүн өгөгдөлд ой бэлдэх, гарааны утга олгох зорилготойгоор уг функцийг тодорхойлдог. Объектийг олон янзаар байгуулах боломж олгоно. Мөн утга буцаагдаггүй бөгөөд void гэсэн түлхүүр үгийг ч хэрэглэдэггүй. Объектийг байгуулах үед тохирох байгуулагчийг нь систем нэг удаа дуудаж хэрэглэдэг. Харин нэгэнт байгуулагдсан объектэд дуудаж хэрэглэхгүй. Классд ямар нэг байгуулагч функцийг илээр тодорхойлж өгөөгүй тохиолдолд C++ compiler хамгийн бага хэмжээ бүхий анхдагч фунцийг дуудна.
- **Устгагч функц**
  - Ямар нэгэн байгуулагч эсвэл устгагч функцгүйгээр объект үүсч эсвэл устахгүй. Цаашид хэрэглэгдэхгүй объектыг устгах түүний эзэмжшиж байсан ойг чөлөөлөх үүрэгтэй. Public шинжтэй байна. Классд ямар нэг устгагч функцийг илээр тодорхойлж өгөөгүй тохиолдолд C++ compiler хамгийн бага хэмжээ бүхий устгагч фунцийг дуудна. Классын нэртэй ижил нэртэй байна.
- **Зарлах, тодорхойлох**
  - Байгуулагч функц тодорхойлох: Дараах загвараар тодорхойлно.

```
class_name () {
    strcpy (data0, “”);
    data1 = 0;
    data2 = 0;
    …
}
```

- Устгагч функц тодорхойлох: Зөвхөн өмнөө ~ тэмдэгтэй бичигдэгээрээ ялгагддаг. Employee классын устгагч функцийг ~Employee () гэж зарлаж, дараа нь тодорхойлно. Жишээ нь:

```
~employee () {
    Cout << "\nObject" << "emp" << k << "is destroyed.";
    k--;
    getch();
}
```

3.2 Функц дахин тодорхойлох гэж юу вэ? Жишээгээр тайлбарал.

- **Функц дахин тодорхойлох**
  - С++ хэлэнд ижил нэртэй, нэгэн төрлийн боловч авах аргументийн тоо, төрөл, дараалал нь ялгаатай байх олон янзын фунцийг тодорхойлох боломжтой. Ингэж нэр, төрөл нь ижил олон функц тодорхойлох энэхүү үйлийг функц дахин тодорхойлох гэх бөгөөд тухайн олон тодорхойлогдож буй функцийг дахин тодорхойлогдох функц гэнэ. Дахин тодорхойлох бол тухайн байдалд бичдэс нь ялгаатай байхаар тодорхойлогдох ижил нэртэй функцүүдийн чавдар юм.
- **Жишээ нь**

```
#include <iostream.h>
#include <conio.h>
int sum(int, int); // int утга буцаадаг, 2 аргументтэй sum функцийг зарлаж байна.
int sum(int, int, int); // sum функцийг 3 аргументтэйгээр дахин зарлаж байна.

int main(){
        clrscr();
        cout<<sum(7, 3, 4)<<endl;
        cout<<sum(7, 3, 4)<<endl;
        getch();
}

int sum(int a, int b){ // sum(int, int) тодорхойлолт
        return  a + b;
}

int sum(int a, int b, int c ){ // sum(int, int) тодорхойлолт
        return a + b + c;
}
```

3.3 Динамик санах ой (new, delete оператор ашигласан) болон байгуулагч, устагч функцийг хэрхэн хамтад нь ашиглах вэ?

- **Динамик санах ой болон байгуулагч, устагч функцийг хамтад нь ашиглах**
  - Байгуулагч функцийн үндсэн үүрэг нь гишүүн өгөгдөлд ой бэлдэх, гарааны утга олгох тул динамик ой нөөцлөх new операторыг ашиглаж болно. Ингэсэн тохиолдолд цаашид хэрэглэгдэхгүй болсон объектыг устгах, түүний эзэмжшиж байсан ойг чөлөөлөх үндсэн үүрэг бүхий устгагч функцэд байгуулагч функцэд нөөцөлсөн ойг delete операторыг ашиглан чөлөөлөх шаардлагатай. Учир нь тухайн ойг устгаагүйгээс үүдэж ойн цоорхой үүсэж, үр ашиггүй ашиглалт бий болох юм.

- **Жишээ нь**

```
class Employee
{
        private :
          char char * name;
          int basicpay;
          int allowance;
        public:
          Employee(char *n = “”, int b = 0, int a = 0);
          ~Employee();
}

Employee::Employee(char *n, int b, int a)
{
        name = new char(strlen(n) + 1); //n хувьсагчид түүний уртын хэмжээгээр ой
        //нөцөөлөх
        strcpy(name, n);
        basicpay = b;
        allowance = a;
}

Employee::~!Employee()
{
        cout << name << endl;
        delete name; // нөөцөлсөн ойг чөлөөлөх
}
```

4. ХЭРЭГЖҮҮЛЭЛТ

1. Lab03-д тодорхойлсон класст анхдагч болон параметртэй байгуулагч нэмж тодорхойлох ба тодорхойлохдоо байгуулагч функцийн үүргийг сайтар хэрэгжүүлсэн байна.

```
class Employee {
private:
   int empid;
   char *name;
   char *position;
   float wtime;
public:
   Employee(); //Андагч байгуулагч функц зарлав.
   Employee(char *n, char *p, int e, float wt); //Параметртэй байгуулагч функц зарлав.
   ~Employee();
   void getData(void);
   void showData();
   float calcSalary();
   float ceoCalcSalary();
   int incWorkTime(float inc);
};
```

```
Employee::Employee() //Анхдагч байгуулагч функц тодорхойлов. Ямар нэг
параметргүйгээр тухайн классын объект байгуулах үед дуудагдана.
{
   empid = 0; //empid - д анхны утга оноов.
   name = new char(1); // name хувьсагчид char төрлийн 1 байт ой нөөцлөв.
   strcpy(name, ""); // 1 byte урттай нөөцөлсөн ойд хоосон буюу null утга оноов.

   position = new char(9);
   strcpy(position, "Employee");
   wtime = 0;
}

Employee::Employee(char* n, char* p, int e, float wt) //Параметртэй байгуулагч функц
тодорхойлов.
{
   name = new char(strlen(n) + 1); // name хувьсагчид n урт дээр нэмэх 1 ойг нөөцлөв.
   strcpy(name, n); // Нөөцөлсөн ойд n параметрийн утгыг олгов.
   position = new char(strlen(p) + 1);
   strcpy(position, p);
   empid = e;
   wtime = wt;
}

Employee::~Employee()
{
   cout<<"\n deleted " << name <<endl;
   delete name;
   delete position;
}
```

2. Lab03-д тодорхойлсон класст устгагч функц тодорхойлж устгагч функц хэзээ дуудагдаж байгааг туршилтад үндэслэн хариул.

```
int main() //Туршилт №1
{
   Employee e1, e2, e3("Bagaa", "Software", 123, 10000), e4("Erdene", "CS", 322, 12000);

   //e1, e2 объектууд нв анхдагч байгуулаг функцээр байгуулагдана.
   //e3, e4 объектууд нь параметртэй байгуулагч функцээр байгуулагдана.

   e1.showData(); // e1 - ийн data - г харах
   e2.showData(); // e2 - ийн data - г харах
   e3.showData(); // e3 - ийн data - г харах
   e4.showData(); // e4 - ийн data - г харах

   // Дээрх объектийг мэдээлэл харуулах үйлдлүүд хийгпэж дууссаны дараа ямар нэг
үйлдэл хэрэгжүүлээгүзй тул
   // устгагч функц автоматаар дуудагдах бөгөөд сүүлд үүссэн объект нь түрүүлж устах
юм.
   return 0;
}
```

- **Туршилтийн үр дүн:**

```
Employee ID: 0
Name:
Position: Employee
Work time: 0

Employee ID: 0
Name:
Position: Employee
Work time: 0

Employee ID: 123
Name: Bagaa
Position: Software
Work time: 10000

Employee ID: 322
Name: Erdene
Position: CS
Work time: 12000


 deleted Erdene

 deleted Bagaa

 deleted

 deleted

Process returned 0 (0x0)   execution time : 0.010 s
```

```
int main(void) //Туршил №2
{
   Employee e1, e3("Bagaa", "Software", 123, 10000);
   //e1, e2 объектууд нв анхдагч байгуулаг функцээр байгуулагдана.
   //e3, e4 объектууд нь параметртэй байгуулагч функцээр байгуулагдана.
   e1.showData(); // e1 - ийн data - г харах
   e3.showData(); // e3 - ийн data - г харах

   Employee e2, e4("Erdene", "CS", 322, 12000);
   e2.showData(); // e2 - ийн data - г харах
   e4.showData(); // e4 - ийн data - г харах

   return 0;

   // Дээрх объектийг мэдээлэл харуулах үйлдлүүд хийгпэж дууссаны дараа ямар нэг объекттой харьцсан үйлдэл хэрэгжүүлээгүй буюу програм дуусах тул
   // устгагч функц автоматаар дуудагдах бөгөөд сүүлд үүссэн объект нь түрүүлж устах юм. Өөрөөр хэлбэл програм бүрэн дууссан үед
   // компайлер устгагч функцийг дуудаж, хамгийн сүүлд үүссэн объектоос эхлэн эзэмшиж буй ойг нь чөлөөлнө.
}
```

- **Туршилтийн үр дүн**

```
Position: Employee
Work time: 0

Employee ID: 123
Name: Bagaa
Position: Software
Work time: 10000

Employee ID: 0
Name:
Position: Employee
Work time: 0

Employee ID: 322
Name: Erdene
Position: CS
Work time: 12000


 deleted Erdene

 deleted

 deleted Bagaa

 deleted

Process returned 0 (0x0)   execution time : 0.009 s
Press any key to continue.
```

3. Lab03-д тодорхойлсон классын захирлын цалин бодох функцийг private хандалтын түвшинтэй болгож өөрчил. Дараа нь цалин бодох функц дотор албан тушаал нь захирал байвал захирлын цалин бодох функцийг цалин бодох функц дотор дуудаж захирлын цалинг бодно. Энд гишүүн функц дотроос гишүүн функц дуудах үйлдлийг хийнэ.

```
class Employee {
private:
   int empid;
   char *name;
   char *position;
   float wtime;
   float ceoCalcSalary();
public:
   Employee(); //Андагч байгуулагч функц зарлав.
   Employee(char *n, char *p, int e, float wt); //Параметртэй байгуулагч функц зарлав.
   ~Employee();
   void getData(void);
   void showData();
   float calcSalary();
   int incWorkTime(float inc);
};
```

```
float Employee::calcSalary()
{
   if(strcmp(position, "CEO") == 0){
      return wtime*pricePerHour + ceoCalcSalary();
   }
   return wtime*pricePerHour;
}

float Employee::ceoCalcSalary()
{
   //if(strcmp(position, "CEO") == 0)
   //уг функцийг хандалттай болгосон учраас энд заавал захирал мөн эсэхийг шалгах
   //шаардлагагүй болно.
   return wtime*2*pricePerHour;
}
```

4. Мөн гишүүн өгөгдөл бүрийг private хандалтын түвшинтэй болгож лекц дээр заасан set болон get функц бичиж утга оноож, утгыг буцааж авна.

```
class Employee {
private:
   int empid; // Ажилтаны дугаарыг private төлөвтэйгээр зарлав.
   char *name; //Ажилтны нэрийг заагч төрлөөр, ptivate төлөвтэйгээр зарлав.
   char *position; //Ажилтны албан тушаалыг заагч төрлөөр, ptivate төлөвтэйгээр зарлав.
   float wtime; //Ажилтны ажилласан цагийг зарлав.
   float ceoCalcSalary(); //Захиралын цалин бодох функцийг private төлөвтэйгээр зарлав.
public:
   Employee(); //Андагч байгуулагч функц зарлав.
   Employee(char *n, char *p, int e, float wt); //Параметртэй байгуулагч функц зарлав.
   ~Employee(); // Устгагч функц зарлав
   void setData(); //гишүүн өгөгдөлд гаднаас хандаж, өөрчлөлт оруулах функц
   void getData(); //гишүүн өгөгдлийн утгыг гаднаас хандаж, авах функц
   float calcSalary(); //Цалин бодох функцийг зарлав.
   int incWorkTime(float inc); //Ажилласан цагийг нэмэгдүүлэх функцийг зарлав.
};
void Employee::setData(void) // set фунцкцийг тодорхойлов.{
   cout<<"Enter employee ID: ";
   cin>>empid;
   cout<<"Enter name:  ";
   cin>>name;
   cout<<"Enter position: ";
   cin>>position;
   cout<<"Enter work time: ";
   cin>>wtime;
}
void Employee::getData() // get функцийг тодорхойлов.{
   cout<<"Employee ID: "<<empid<<endl;
   cout<<"Name: "<<name<<endl;
   cout<<"Position: "<<position<<endl;
   cout<<"Work time: "<<wtime<<endl<<endl;
}
```

5. Ажилчин классын хүснэгт үүсгээд гараас хэд хэдэн ажилчны утга онооно.

```
int main()
{
   int n;
   cout<<"Enter count of employee: ";
   cin>>n;

   Employee e[n];

   cout<<"Enter data of each employee: "<<endl;
   for(int i = 0; i < n; i++)
   {
      cout<<"\tEmployee No"<<i + 1<<":\n";
      e[i].setData();
   }

   return 0;
}
```

```
Enter count of employee: 3
Enter data of each employee:
        Employee No1:
Enter employee ID: 121
Enter name:  Bagaa
Enter position: SE
Enter work time: 1200
        Employee No2:
Enter employee ID: 122
Enter name:  Galaa
Enter position: CS
Enter work time: 1500
        Employee No3:
Enter employee ID: 123
Enter name:  Bandia
Enter position: IT
Enter work time: 1000
```

6. Утга оноосон хүснэгтийг цалингаар нь эрэмбэлнэ.

```
   int n;
   cout<<"Enter count of employee: ";
   cin>>n;

   Employee e[n];

   cout<<"Enter data of each employee: "<<endl;
   for(int i = 0; i < n; i++)
   {
      cout<<"\tEmployee No"<<i + 1<<":\n";
      e[i].setData();
   }
```

```
  for(int i = 1; i < n; i++)
  {
     Employee key = e[i];
     float key_sal = e[i].calcSalary();
     int j = i - 1;
     while(j >= 0 && key_sal < e[j].calcSalary())
     {
        e[j + 1] = e[j];
        j--;
     }

     e[j + 1] = key;
  }

  cout<<"\n\nEmloyees were sorted: "<<endl;
  for(int i = 0; i < n; i++)
  {
     e[i].getData();
  }
```

```
Emloyees were sorted:
Employee ID: 123
Name: Bandia
Position: IT
Work time: 1000

Employee ID: 121
Name: Bagaa
Position: SE
Work time: 1200

Employee ID: 122
Name: Galaa
Position: CS
Work time: 1500
```

5. **ДҮГНЭЛТ**

Энэхүү лабораторын ажлын асуултуудын хүрээнд Employee буюу ажилтан гэх класс хөгжүүлсэн. Тухайн классын гишүүн өгөгдлүүдийг private хэлбэрээр, захирлын цалин тооцох гишүүн функцээс бусад гишүүн функцийг public төлөвөөр ханддаг байхаар зохион байгуулсан. Ингэснээр тухайн классын гишүүн өгөгдлийг заавал уг классын гишүүн функцийн тусламжтайгаар өөрчилж, эсвэл авах бөгөөд цаашлаад гишүүн өгөгдөл гадны санамсаргүй болон санаатай халдлагаас хамгаалагдах юм. Мөн цалин бодох функцийг public төлөвөөр зарласан үед шууд гаднаас хандах боломж үүсэх бөгөөд хэрэв тухайн функц дотор тухайн параметрээр орж орж буй объект нь захирал мөн эсэхийг шалгадаг шалгуур байхгүй бол дурын ажилтан захирлын цалин авах үүсэх тул уг функцийг private – ээр зарлаж өгөх шаардлагатай. Уг функцийг классын гишүүн функц дотор дуудан ашиглаж байгаа учир private байхад ямар нэг асуудал үүсэхгүй. Үүнээс гадна классын тэмдэг мөрөн хувьсагчдын заагчаар зарлаж ашигласан нь санах ойг үр ашигтай ашиглах боломжийг олгож буй явдал юм.

6. **АШИГЛАСАН МАТЕРИАЛ**

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

7. **ХАВСРАЛТ**

```
#include <iostream>
#include <string.h>
#define pricePerHour 5000
using namespace std;

class Employee {
private:
   int empid; // Ажилтаны дугаарыг private төлөвтэйгээр зарлав.
   char *name; //Ажилтны нэрийг заагч төрлөөр, ptivate төлөвтэйгээр зарлав.
   char *position; //Ажилтны албан тушаалыг заагч төрлөөр, ptivate төлөвтэйгээр зарлав.
   float wtime; //Ажилтны ажилласан цагийг зарлав.
   float ceoCalcSalary(); //Захиралын цалин бодох функцийг private төлөвтэйгээр зарлав.
public:
   Employee(); //Андагч байгуулагч функц зарлав.
   Employee(char *n, char *p, int e, float wt); //Параметртэй байгуулагч функц зарлав.
   ~Employee(); // Устгагч функц зарлав
   void setData(); //гишүүн өгөгдөлд гаднаас хандаж, өөрчлөлт оруулах функц
   void getData(); //гишүүн өгөгдлийн утгыг гаднаас хандаж, авах функц
   float calcSalary(); //Цалин бодох функцийг зарлав.
   int incWorkTime(float inc); //Ажилласан цагийг нэмэгдүүлэх функцийг зарлав.
}; float Employee::calcSalary() //цалин бодох функцийг тодорхойлов.
{
   if(strcmp(position, "CEO") == 0){
      return wtime*pricePerHour + ceoCalcSalary();
   }
   return wtime*pricePerHour;
}

float Employee::ceoCalcSalary() // захирлын цалин бодох функцийг тодорхойлов.
{
   //if(strcmp(position, "CEO") == 0)
   //уг функцийг хандалттай болгосон учраас энд заавал захирал мөн эсэхийг шадгах
   //шаардлагагүй болно.
   return wtime*pricePerHour;
}

Employee::Employee() //Анхдагч байгуулагч функц тодорхойлов. Ямар нэг параметргүйгээр тухайн классын объект байгуулах үед дуудагдана.
{
   empid = 0; //empid - д анхны утга оноов.
   name = new char(1); // name хувьсагчид char төрлийн 1 байт ой нөөцлөв.
   strcpy(name, ""); // 1 byte урттай нөөцөлсөн ойд хоосон буюу null утга оноов.

   position = new char(9);
   strcpy(position, "Employee");
   wtime = 0;
}

Employee::Employee(char* n, char* p, int e, float wt) //Параметртэй байгуулагч функц тодорхойлов.
{
```

```
name = new char(strlen(n) + 1); // name хувьсагчид n урт дээр нэмэх 1 ойг нөөцлөв.
   strcpy(name, n); // Нөөцөлсөн ойд n параметрийн утгыг олгов.
   position = new char(strlen(p) + 1);
   strcpy(position, p);
   empid = e;
   wtime = wt;
}

Employee::~Employee() //устгагч функцийг тодорхойлов.
{
   cout<<"\n deleted " << name <<endl;
   delete name;
   delete position;
}

int Employee::incWorkTime(float inc) //ажилласан цаг нэмэгдүүлэх функцийг
тодорхойлов.
{

   if(0 < inc && inc <= 24){
      wtime += inc;
      return 1;
   }

   return 0;
}

int main()
{
   int n;
   cout<<"Enter count of employee: ";
   cin>>n;

   Employee e[n];

   cout<<"Enter data of each employee: "<<endl;
   for(int i = 0; i < n; i++)
   {
      cout<<"\tEmployee No"<<i + 1<<":\n";
      e[i].setData();
   }


   for(int i = 1; i < n; i++)
   {
      Employee key = e[i];
      float key_sal = e[i].calcSalary();
      int j = i - 1;
      while(j >= 0 && key_sal < e[j].calcSalary())
      {
         e[j + 1] = e[j];
```

```
            j--;
        }

        e[j + 1] = key;
    }

    cout<<"\n\nEmloyees were sorted: "<<endl;
    for(int i = 0; i < n; i++)
    {
        e[i].getData();
    }

    return 0;
}
```