Лабораторийн ажил №6

Н. Мөнхжин

ХШУИС, МКУТ, Компьютерийн ухаан, nyamkamunhjin@gmail.com

1. ОРШИЛ

Энэ лабораторийн ажлаар Оюутан Объект загварыг Классын тусламжтайгаар үүсгэж шаардлагатай байгуулагч функц, устгагч функц, гишүүн өгөгдөл, гишүүн функцийг тодорхойлж тулгарах асуудлуудыг шийдэж гүйцэтгэнэ.

2. ЗОРИЛГО

Оюутан класс-д:

1. Гишүүн өгөгдлийг тодорхойлох.
2. Байгуулагч, устгагч функцуудыг тодорхойлох.
3. Гишүүн функцуудыг тодорхойлох.
4. Оюутан объектууд хадгалах хүснэгт үүсгэх.
5. Хүснэгтийг нэрээр, күрсээр эрэмблэх.

3. ОНОЛЫН СУДАЛГАА

3.1. Хуулагч байгуулагч

Ямар нэгэн Классын хуулагч байгуулагч функц нь ижил төрлийн Классын объектын гишүүн өгөгдлүүдын утгыг өөрийн гишүүн өгөгдөл дээр хуулан оноох юм. Ашиглахдаа = утга оноох оператор хэрэглэж болдог. Гэхдээ зөвхөн Объектыг анх зарлаж байх үед ашиглана. Student a = b; гэх мэт.

3.2 Статик төлөв

Гишүүн өгөгдлийг статик төлөвтэй тодорхойлсноор тухайн программыг төгсөх хүртэл санах ойд нөөцлөгдөн чөлөөлөгдөхгүй байж байдаг. Класс-д тодорхойлогдсон статик гишүүн өгөгдөл нь ижил Классын объектуудаас хандах боломжтой байдаг. Харин статик гишүүн функц зарлаж өгөхөд объектыг тодорхойлоогүй үед дуудаж болдог. Гэхдээ зөвхөн өөр статик төлөвтэй гишүүн өгөгдөл, гишүүн функцуудыг дуудах боломжтой.

3.3 This заагч

This заагчийн тусламжтайгаар Класс доторх гишүүн өгөгдөл, функцуудрүү хандах боломжтой. This заагч нь тухайн объектын заагчийн хаягийг хадгалж байдаг тул ингэж

хандах боломжтой. this->name, this->getName(), this->id гэх мэт.

4. ХЭРЭГЖҮҮЛЭЛТ

1.Student Классын тодорхойлолт:

```
class Student {
private:
   char *studentId = NULL;
   float gpa;
   const static int counter = 1;
public:
   char *name = NULL;
   int year;

   // getters
   char* getStudentId();
   char* getName();
   int getYear();
   float getGpa();
   int getCounter();

   // setters
   void setStudentId(char*);
   void setName(char*)
   void setYear(int);
   void setGpa(int);
   void copy(Student);

   // Constructors

   // Default Constructor
   Student();
   // Parameterized Constructor
   Student(char*, char*, int, float);
   // Copy Constructor
   Student(const Student &);
  // Destructor
   ~Student()
```

```
// Comparing
bool greatThan(Student &);
bool lessThan(Student &);
bool isEqual(Student &);
bool before(Student &);
bool after(Student &);
```

2. Байгуулагч функц.

Энэ жишээгээр хуулагч байгуулагч функцийг харуулав.

Санах ойн цоорхой үүсгэхгүй тулд хуулж байгаа объектын үсэгний дараалалд тохируулан санах ой нөөцлөх шаардлагатай.

```
Student(const Student &copyStudent) {
        if(this->studentId != NULL) delete this->studentId;
        if(this->name != NULL) delete this->name;

        this->studentId = new char[strlen(copyStudent.studentId)];
        this->name = new char[strlen(copyStudent.name)];

        strcpy(this->studentId, copyStudent.studentId);
        strcpy(this->name, copyStudent.name);
        this->year = copyStudent.year;
        this->gpa = copyStudent.gpa;
    }
```

3. Оюутан объектуудыг хооронд нь харьцуулах гишүүн функц.

Курсээр нь харьцуулах функц. Хэрэв харьцуулагдаж байгаа оюутан объект харьцуулж байгаа оюутан объектоос күрс илүү бол 0 эсвэл 1-ийг буцаана.

```
bool greatThan(Student &student) {
    return this->year > student.getYear();
}
```

Нэрээр нь харьцуулах. Strcmp функц нь үсэг үсгээр нь харьцуулдаг функц юм. Харгалзал үсгүүдыг хооронд нь харьцуулж их байвал 1, бага байвал -1, үг ижил байвал 0-ийг буцаана.

```
bool before(Student &student) {
```

```
        return strcmp(this->name, student.getName()) > 0;
    }
```

4. Олон оюутныг хүснэгтэнд оруулахын тулд c++ хэлний ‘vector’ өгөгдлийн бүтцийг ашигласан. Энэ бүтэц нь хүснэгттэй төстэй бөгөөд давуу тал нь өгөгдөл нэмэх үед хүснэгтийн хэмжээ бас дагаад ихсэж байдаг.

```
vector<Student> studentList;
studentList.push_back(Student("16b1", "c", 4, 3.1));
studentList.push_back(Student("ewqe", "b", 1, 4));
```

5. Оюутнуудыг эрэмблэхийн тулд Оюутан Класс дотор тодорхойлогдсон `void copy(Student), Student(const Student &), bool after(Student &)` функцүүдийг ашиглав. Эрэмблэх алгоритмийг quickSort-оор шийдэж өгөв. Харин Оюутан объектыг хооронд солихын тулд `void swapStudent(Student &a, Student &b)` тодорхойлж өгөв.

```
void sortByYear(vector<Student> &list, int start, int end) {
    if(start >= end) return;

    Student pivot = list[end - 1];
    int index = start;

    for(int i = start; i < end - 1; i++) {
        if(list[i].getYear() < pivot.getYear()) {
            swapStudent(list[i], list[index]);
            index++;
        }
    }
```

5. ДҮГНЭЛТ

Энэ лабораторийн ажлаар Оюутан объектыг Классаар загварчлан хийж гүйцэтгэв. Нэр болон ID-г динамик санах ойгоор нөөцөлж өгснөөр анх зарласан хэмжээ оруулж буй утганд хэтэрхий бага байх эсвэл том санах ой зарлаж өгснөөр илүү санах ойн зайг эзэлж байгаа асуудлуудыг шийдэв. Нэмэлт функцуудыг тодорхойлж өгснөөр ямар ч асуудалгүй объектуудыг хооронд нь харьцуулах, солих боломжтой болж байна.

6. АШИГЛАСАН МАТЕРИАЛ

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

[1] Хуулагч байгуулагч /хуудас-97/

7. ХАВСРАЛТ

Student.cpp

```
#include<iostream>
#include<string.h>


class Student {
public:
   char *name = NULL;
   int year;


   // getters
   char* getStudentId() {
       return this->studentId;
   }

   char* getName() {
       return this->name;
   }

   int getYear() {
       return this->year;
   }

   float getGpa() {
       return this->gpa;
   }

   int getCounter() {
       return this->counter;
```

```
}

// setters
void setStudentId(char *studentId) {
    delete this->studentId;
    this->studentId = new char[strlen(studentId)];
    strcpy(this->studentId, studentId);
}

void setName(char *name) {
    delete this->name;
    this->name = new char[strlen(name)];
    strcpy(this->name, name);
}

void setYear(int year) {
    if(year != 0) this->year = year;
}

void setGpa(int gpa) {
    if(gpa >= 0.0 && gpa <= 4.0) this->gpa = gpa;
}

void copy(Student student) {
    delete this->studentId;
    delete this->name;

    this->studentId = new char[strlen(student.getStudentId())];
    this->name = new char[strlen(student.getName())];
    strcpy(this->studentId, student.getStudentId());
    strcpy(this->name, student.getName());
    this->year = student.getYear();
    this->gpa = student.getGpa();
}


// Constructors

// Default Constructor
```

```
Student() {
    this->name = new char;
    this->studentId = new char;
    this->year = 1;
    this->gpa = 0;

}

// Parameterized Constructor
Student(char *studentId, char *name, int year, float gpa) {
    if(this->studentId != NULL) delete this->studentId;
    if(this->name != NULL) delete this->name;

    this->studentId = new char[strlen(studentId)];
    strcpy(this->studentId, studentId);

    this->name = new char[strlen(name)];
    strcpy(this->name, name);

    if(year != 0) this->year = year;
    else this->year = 1;

    if(gpa >= 0.0 && gpa <= 4.0) this->gpa = gpa;
    else this->gpa = 0;
}

// Copy Constructor
Student(const Student &copyStudent) {
    if(this->studentId != NULL) delete this->studentId;
    if(this->name != NULL) delete this->name;

    this->studentId = new char[strlen(copyStudent.studentId)];
    this->name = new char[strlen(copyStudent.name)];

    strcpy(this->studentId, copyStudent.studentId);
    strcpy(this->name, copyStudent.name);
    this->year = copyStudent.year;
    this->gpa = copyStudent.gpa;
}
```

```
  // Destructor
   ~Student() {
       // std::cout << "Deleting: " << this->name << std::endl;
       delete this->name;
       delete this->studentId;
   }

   // Comparing
   bool greatThan(Student &student) {
       return this->year > student.getYear();
   }

   bool lessThan(Student &student) {
       return this->year < student.getYear();
   }

   bool isEqual(Student &student) {
       return this->year == student.getYear();
   }

   bool before(Student &student) {
       return strcmp(this->name, student.getName()) > 0;
   }

   bool after(Student &student) {
       return strcmp(this->name, student.getName()) < 0;
   }



private:
   char *studentId = NULL;
   float gpa;
   const static int counter = 1;

};
```

Main.cpp

```
--------------------------------------------------------------------------------
#include <iostream>
#include <string.h>
#include <vector>
#include "Student.cpp"
using namespace std;

int countStudents(vector<Student> list) {
   int count = 0;
   for(int i = 0; i < list.size(); i++) {
       count += list[i].getCounter();
   }

   return count;
}

void swapStudent(Student &a, Student &b) {
   // copying a
   Student temp = a;

   // copying fields of b to a
   a.copy(b);

   // copying field of (copy a) to b
   b.copy(temp);

}



void sortByName(vector<Student> &list, int start, int end) {
   if(start >= end) return;


   Student pivot = list[end - 1];
   int index = start;

   for(int i = start; i < end - 1; i++) {
       if(list[i].after(pivot)) {
```

```
            swapStudent(list[i], list[index]);
            index++;
        }
    }
    swapStudent(list[index], list[end - 1]);

    sortByName(list, start, index);
    sortByName(list, index + 1, end);
}

void sortByYear(vector<Student> &list, int start, int end) {
    if(start >= end) return;

    Student pivot = list[end - 1];
    int index = start;

    for(int i = start; i < end - 1; i++) {
        if(list[i].getYear() < pivot.getYear()) {
            swapStudent(list[i], list[index]);
            index++;
        }
    }
    swapStudent(list[index], list[end - 1]);

    sortByYear(list, start, index);
    sortByYear(list, index + 1, end);
}

void printStudent(vector<Student> list) {
    printf("id\tname\tyear\tgpa\n");
    printf("-----------------------------------------\n");
    for(int i = 0; i < list.size(); i++) {
        printf("%s\t%s\t%d\t%f\n", list[i].getStudentId(), list[i].getName(),
list[i].getYear(), list[i].getGpa());
    }
    printf("-----------------------------------------\n");
}

int main() {
```

```
vector<Student> studentList;
studentList.push_back(Student("16b1", "c", 4, 3.1));
studentList.push_back(Student("ewqe", "b", 1, 4));
studentList.push_back(Student("qwew", "a", 2, 2.5));
studentList.push_back(Student("qwds", "e", 3, 1));
studentList.push_back(Student("qzxc", "d", 3, 3.3));
printStudent(studentList);

sortByName(studentList, 0, studentList.size());
printStudent(studentList);

sortByYear(studentList, 0, studentList.size());
printStudent(studentList);

cout << countStudents(studentList) << endl;
```