# Оюутан классыг байгуулж өгөгдсөн даалгаврыг гүйцэтгэх лабораторийн даалгавар

# (Лаборатори №7)

Г. Саруул (20B1NUM2095)

ХШУИС, МКУТ-ийн Компьютерийн ухааны хөтөлбөрийн

2-р түвшиний оюутан

# 1. ОРШИЛ

Олон оюутны мэдээллийг хүснэгтэд хадгалж, оюутны мэдээллийг ашиглан эрэмбэлсэн.

2.ЗОРИЛГО

Лекцийн хичээл дээр үзсэн зүйлээ бататгаж, өгөгдсөн даалгавруудыг гүйцэтгэх.

Хийх даалгавар:

-олон оюутан үүсгэж объектон хүснэгтэд хадгал.

-хүснэгт дэх объектуудыг нэрээр нь эрэмбэл.

-хүснэгт дэх объектуудыг курсээр нь эрэмбэл.

-статик гишүүн өгөгдөл ашиглан хэдэн оюутан үүссэн байгааг харуул.

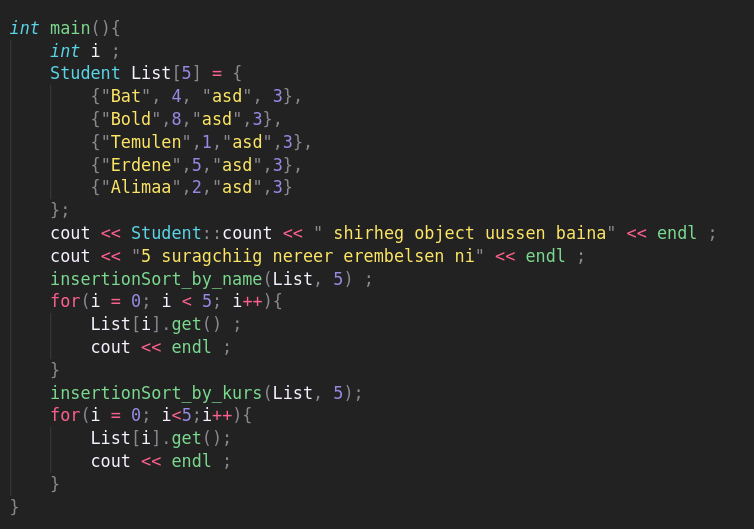
# 3. ОНОЛЫН СУДАЛГАА

## 3.1 Санах ойн цоорхой үүсгэхгүй байхын тулд хуулагч функц, хуулагч функц ашиглах хэрэгтэй.

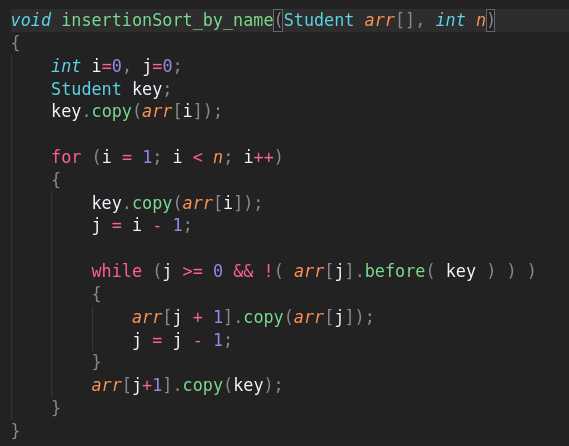
3.2 This хаяган хувьсагч нь функцийг дуудаж байгаа объектийн хаягийг заадаг хувьсагч юм. This хаяган хувьсагчийг ашиглах нь кодыг уншихад илүү хялбар болгох буюу дуудсан объектын гишүүн өгөгдлийг ялгахад амар болдог.

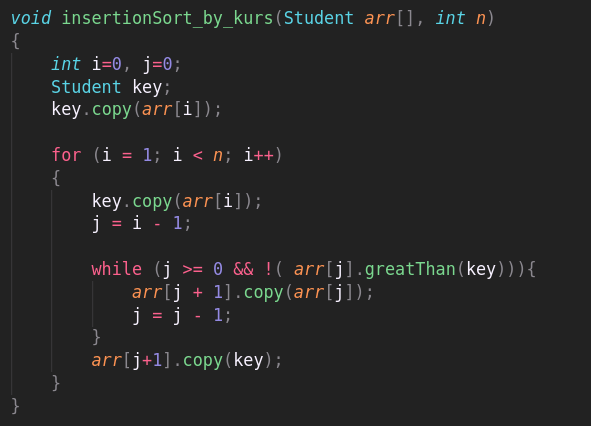
4. ХЭРЭГЖҮҮЛЭЛТ

1. static гишүүн өгөгдөл ашиглан хэдэн оюутан үүссэн байгааг харуулсан.
2. Main.cpp файлд объектоос бүрдсэн List хүснэгтийн эрэмбэлэхийн тулд insertSort\_by\_name, insertSort\_by\_kurs гэдэг 2 функцийг тодорхойлж ашигласан. Энэ 2 функцийн жиших хэсэгт Class дотор тодорхойлсон Student::before(), Student::greatThan() гэсэн гишүүн функцуудыг ашигласан.



Эрэмбэлэх функцуудыг тодорхойлсон нь:





Student.cpp

#include "Student.h"

#include<iostream>

#include<string.h>

using *namespace* std;

*int* Student::count = 0;

Student::Student(){

*this*->name = new *char*[20] ;

*this*->name = NULL ;

*this*->id = new *char*[11] ;

*this*->id = NULL ;

*this*->kurs = 1 ;

*this*->gpa = 0 ;

Student::count++ ;

}

Student::Student(*char* \**name*, *unsigned* *int* *kurs*, *char* \**id*, *float* *gpa*){

*this*->name = new *char*[strlen(*name*) + 1] ;

strcpy(*this*->name, *name*) ;

*this*->kurs = *kurs* ;

*this*->id = new *char*[strlen(*id*) + 1] ;

strcpy(*this*->id, *id*) ;

*this*->gpa = *gpa* ;

Student::count++ ;

}

Student::Student(*const* Student &*src*) {

*this*->name = new *char*[strlen(*src*.name)+ 1] ;

strcpy(*this*->name, *src*.name) ;

*this*->kurs = *src*.kurs ;

*this*->id = new *char*[strlen(id) + 1] ;

strcpy(*this*->id, id) ;

*this*->gpa = *src*.gpa ;

Student::count++ ;

}

*void* Student::copy(*const* Student *&src*){

delete *this*->name ;

delete *this*->id ;

*this*->name = new *char*[strlen(*src*.name)+ 1];

strcpy(*this*->name, *src*.name) ;

*this*->id = new *char*[strlen(*src*.id) + 1] ;

strcpy(*this*->id, *src*.id);

*this*->gpa = *src*.gpa ;

*this*->kurs = *src*.kurs ;

}

Student::~Student(){

delete *this*->name ;

delete *this*->id ;

*// Student::count-- ;*

}

*// Duudsan objectiin kurs ih uyd*

*bool* Student::greatThan(*const* Student *&a*){

return (*this*->kurs > *a*.kurs) ;

}

*// Duudsan objectiin kurs baga baih uyd*

*bool* Student::lessThan(*const* Student *&a*){

return (*this*->kurs < *a*.kurs) ;

}

*bool* Student::isEqual(*const* Student *&a*){

return (*this*->kurs == *a*.kurs) ;

}

*bool* Student::before(*const* Student *&a*){

return (strcmp(*this*->name, *a*.name) < 0 ) ;

}

*void* Student::get(){

cout << "Name: " << *this*->name << endl ;

cout << "Kurs: " << *this*->kurs << endl ;

cout << "id: " << *this*->id << endl ;

cout << "gpa: " << *this*->gpa << endl ;

}

*void* Student::set(*char* *\*name*, *unsigned* *int* *kurs*, *char* *\*id*, *float* *gpa*){

strcpy(*this*->name, *name*) ;

*this*->kurs = *kurs* ;

strcpy(*this*->id, *id*);

*this*->gpa = *gpa* ;

}

Student.h

*class* Student {

*private:*

*char* \*id ;

*float* gpa ;

*public:*

*char* \*name ;

*unsigned* *int* kurs ;

*static* *int* count ;

Student() ;

Student(*char* *\*name*, *unsigned* *int* *kurs*, *char* *\*id*, *float* *gpa*) ;

Student(*const* Student *&src*) ;

~Student() ;

*void* copy(*const* Student *&src*) ;

*bool* greatThan(*const* Student *&a*) ;

*bool* lessThan(*const* Student *&a*) ;

*bool* isEqual(*const* Student *&a*) ;

*bool* before(*const* Student *&a*) ;

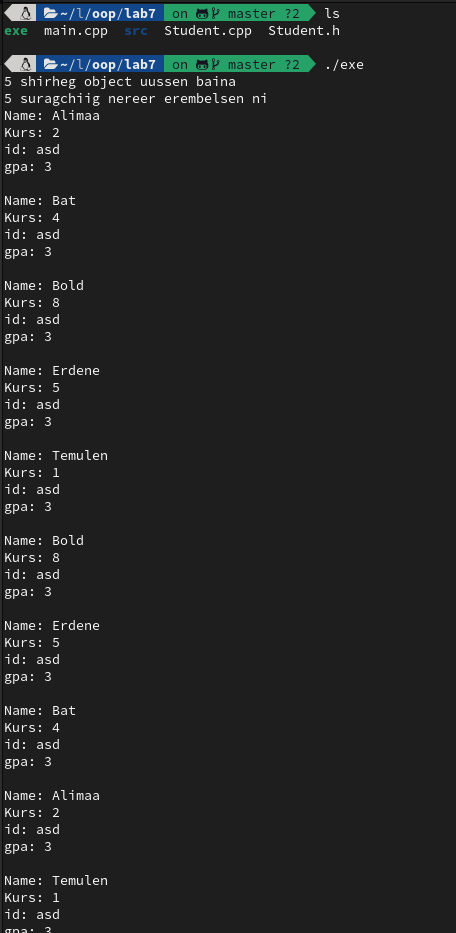
*bool* after(*const* Student *&a*);

*void* get() ;

*void* set(*char* *\*name*, *unsigned* *int* *kurs*, *char* *\*id*, *float* *gpa*);

};

Кодын ажлуулсан байдал:



# 5. ДҮГНЭЛТ

Static хувьсагч буюу class -ийн өгөгдөл зарласнаар хэдэн объект class -аас үүсч байгааг тоолох боломжтой болсон. Хуулагч байгуулагч функц, хуулагч функцууд ашигласнаар санах ойтой холбоотой асуудал үүсэхээс сэргийлсэн.

# 6. АШИГЛАСАН МАТЕРИАЛ

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

2. <http://www.cplusplus.com/doc/tutorial/>

# 7. ХАВСРАЛТ

**Student.h file**

class Student {

private:

char \*id ;

float gpa ;

public:

char \*name ;

unsigned int kurs ;

static int count ;

Student() ;

Student(char \*name, unsigned int kurs, char \*id, float gpa) ;

Student(const Student &src) ;

~Student() ;

void copy(const Student &src) ;

bool greatThan(const Student &a) ;

bool lessThan(const Student &a) ;

bool isEqual(const Student &a) ;

bool before(const Student &a) ;

bool after(const Student &a);

void get() ;

void set(char \*name, unsigned int kurs, char \*id, float gpa);

};

**Student.cpp file**

#include "Student.h"

#include<iostream>

#include<string.h>

using namespace std;

int Student::count = 0;

Student::Student(){

this->name = new char[20] ;

this->name = NULL ;

this->id = new char[11] ;

this->id = NULL ;

this->kurs = 1 ;

this->gpa = 0 ;

Student::count++ ;

}

Student::Student(char \*name, unsigned int kurs, char \*id, float gpa){

this->name = new char[strlen(name) + 1] ;

strcpy(this->name, name) ;

this->kurs = kurs ;

this->id = new char[strlen(id) + 1] ;

strcpy(this->id, id) ;

this->gpa = gpa ;

Student::count++ ;

}

Student::Student(const Student &src) {

this->name = new char[strlen(src.name)+ 1] ;

strcpy(this->name, src.name) ;

this->kurs = src.kurs ;

this->id = new char[strlen(id) + 1] ;

strcpy(this->id, id) ;

this->gpa = src.gpa ;

Student::count++ ;

}

void Student::copy(const Student &src){

delete this->name ;

delete this->id ;

this->name = new char[strlen(src.name)+ 1];

strcpy(this->name, src.name) ;

this->id = new char[strlen(src.id) + 1] ;

strcpy(this->id, src.id);

this->gpa = src.gpa ;

this->kurs = src.kurs ;

}

Student::~Student(){

delete this->name ;

delete this->id ;

// Student::count-- ;

}

// Duudsan objectiin kurs ih uyd

bool Student::greatThan(const Student &a){

return (this->kurs > a.kurs) ;

}

// Duudsan objectiin kurs baga baih uyd

bool Student::lessThan(const Student &a){

return (this->kurs < a.kurs) ;

}

bool Student::isEqual(const Student &a){

return (this->kurs == a.kurs) ;

}

bool Student::before(const Student &a){

return (strcmp(this->name, a.name) < 0 ) ;

}

void Student::get(){

cout << "Name: " << this->name << endl ;

cout << "Kurs: " << this->kurs << endl ;

cout << "id: " << this->id << endl ;

cout << "gpa: " << this->gpa << endl ;

}

void Student::set(char \*name, unsigned int kurs, char \*id, float gpa){

strcpy(this->name, name) ;

this->kurs = kurs ;

strcpy(this->id, id);

this->gpa = gpa ;

}

// **main.cpp File**

#include "Student.h"

#include<iostream>

#include<string.h>

using namespace std;

void insertionSort\_by\_kurs(Student arr[], int n)

{

int i=0, j=0;

Student key;

key.copy(arr[i]);

for (i = 1; i < n; i++)

{

key.copy(arr[i]);

j = i - 1;

while (j >= 0 && !( arr[j].greatThan(key))){

arr[j + 1].copy(arr[j]);

j = j - 1;

}

arr[j+1].copy(key);

}

}

void insertionSort\_by\_name(Student arr[], int n)

{

int i=0, j=0;

Student key;

key.copy(arr[i]);

for (i = 1; i < n; i++)

{

key.copy(arr[i]);

j = i - 1;

while (j >= 0 && !( arr[j].before( key ) ) )

{

arr[j + 1].copy(arr[j]);

j = j - 1;

}

arr[j+1].copy(key);

}

}

int main(){

int i ;

Student List[5] = {

{"Bat", 4, "asd", 3},

{"Bold",8,"asd",3},

{"Temulen",1,"asd",3},

{"Erdene",5,"asd",3},

{"Alimaa",2,"asd",3}

};

cout << Student::count << " shirheg object uussen baina" << endl ;

cout << "5 suragchiig nereer erembelsen ni" << endl ;

insertionSort\_by\_name(List, 5) ;

for(i = 0; i < 5; i++){

List[i].get() ;

cout << endl ;

}

insertionSort\_by\_kurs(List, 5);

for(i = 0; i<5;i++){

List[i].get();

cout << endl ;

}

}