**Лабораторийн сэдэв:** Си програмчлалын хэлийг судлах-3

**Судлах зүйлс:**

* Массив
* Массивыг функц руу дамжуулах
* Функцаас массив буцаах
* Заагч төрөлтэй массив
* Заагч
* Тэмдэгт мөр
* Бүтэц
* Нэгдэл
* Бит талбар
* Тайлбар (comment) ашиглах

**Оноо:** 2 оноо

**Даалгавар-1:** Судлах зүйл дээр бичсэн бүлэг ойлголтуудыг minna.blogmn.net дээрх Си програмчлалын хэлний хичээлийг харна уу. Мөн бусадтайгаа ярилцан (мэдэхгүй ойлголтоо бусдаасаа асуух, мэддэг ойлголтоо бусаддаа тайлбарлах) мэдлэгээ хуваалцах хэрэгтэй. Ярилцснаар мэдлэг баталгаажиж, удаан мартагдана!

**Даалгавар-2:** Дараах бодлогуудын програмыг бичнэ үү.

1. “Тооны машин” програмыг бичнэ үү. Жич: Тооны машин програм нь:

* Ямар ч тооны төрлүүд дээр тооцоолол хийдэг байна.
* Хаалттай илэрхийлэл дээр тооцоолол хийдэг байна.
* Өмнө нь тооцоолсон тооцооллуудыг түүх болгон хадгалдаг байна (History).
* Зэрэг дэвшүүлэх, язгуур авах үйлдлүүдтэй байна.

... гэх мэтчилэнгээр олон үйлдэл хийдэг байх тусмаа сайн.

#include <bits/stdc++.h>

#define pb push\_back

#define mp make\_pair

#define ff first

#define ss second

#define sz(x) (int)(x).size()

#define all(x) (x).begin(),(x).end()

#define PI 3.14159265

using namespace std;

vector <char> box;

int done = 0;

vector<pair<string,string>> history;

inline int isDoneForWithoutBracket (string expression) {

// cout << "Checking : " << expression << '\n';

int leftBracket = 0, rightBracket = 0, operat = 0;

for (int i = 0; i < expression.length(); i++) {

leftBracket += (expression[i] == '(');

rightBracket += (expression[i] == ')');

if (expression[i] == '+' || expression[i] == '-' || expression[i] == '\*' || expression[i] == '/')

operat++;

}

if (operat == 1 && leftBracket == 1 && rightBracket == 1)

return 3;

if (expression[0] == '-') {

bool can = true;

for (int i = 1; i < expression.length(); i++) {

if ((expression[i] >= 48 && expression[i] <= 57) || expression[i] == '.')

continue;

else { can = false; break; }

}

if (can) return 0;

}

for (int i = 0; i < expression.length(); i++)

if (expression[i] == '\*' || expression[i] == '/')

return 1;

for (int i = 0; i < expression.length(); i++)

if (expression[i] == '+' || expression[i] == '-')

return 2;

return 0;

}

template <typename T>

std::string to\_string(T const& value) {

stringstream sstr;

sstr << value;

return sstr.str();

}

inline string calcWithoutBracket (string expression) {

int status = isDoneForWithoutBracket(expression);

cout << "Status " << status << " " << expression << '\n';

switch (status) {

case 1: {

for (int i = 0; i < expression.length(); i++) {

if ((expression[i] == '\*' || expression[i] == '/') && i != 0) {

string left = "", right = "";

int startIndex, endIndex;

bool ok = false;

// Start right side;

if (expression[i + 1] == '(') {

ok = true;

for (int j = i + 2; j < expression.length(); j++) {

if (expression[j] == ')') { endIndex = j + 1; break; }

else right += expression[j];

}

}

if (!ok) {

for (int j = i + 1; j < expression.length(); j++) {

if ((expression[j] >= 48 && expression[j] <= 57) || expression[j] == '.') {

if (j == expression.length() - 1)

endIndex = j;

right += expression[j];

}

else { endIndex = j; break; }

}

}

cout << "RIGHT : " << right << '\n';

// Done right side;

// Start left side;

ok = false;

cout << "EXrpession " << expression << '\n';

if (expression[i - 1] == ')') {

ok = true;

for (int j = i - 2; j >= 0; j--) {

if (expression[j] == '(') { startIndex = j - 1; break; }

else if (j == 0) { left += expression[j]; startIndex = 0; break; }

else left += expression[j];

}

}

cout << "LEFT : " << left << '\n';

if (!ok) {

for (int j = i - 1; j >= 0; j--) {

if (j == 0 && expression[j] == '-') {

startIndex = 0;

left += expression[j];

break;

}

if ((expression[j] >= 48 && expression[j] <= 57) || expression[j] == '.') {

if (j == 0) {

startIndex = 0;

left += expression[j];

break;

}

left += expression[j];

}

else if (j == 0) { startIndex = 0; break; }

else { startIndex = j; break; }

}

}

// cout << "LEFT : " << left << '\n';

// Done left side;

reverse (left.begin(), left.end());

cout << left << " " << right << '\n';

// cout << startIndex << " " << endIndex << "\n";

double num1 = stod(left), num2 = stod(right), ans;

// cout << expression[i] << '\n';

ans = (expression[i] == '\*' ? num1 \* num2 : num1 / num2);

string tmp = to\_string(ans);

// cout << tmp << '\n';

if (startIndex != 0)

if (ans < 0)

tmp = '(' + tmp + ')';

if (startIndex != 0)

for (int j = 0; j <= startIndex; j++) box.push\_back(expression[j]);

for (int j = 0; j < tmp.length(); j++) box.push\_back(tmp[j]);

for (int j = endIndex; j < expression.length(); j++) box.push\_back(expression[j]);

expression = "";

for (int j = 0; j < box.size(); j++) expression += box[j];

// cout << "Now -> " << expression << '\n';

box.clear();

return calcWithoutBracket(expression);

}

}

break;

}

case 2: {

for (int i = 0; i < expression.length(); i++) {

if ((expression[i] == '+' || expression[i] == '-') && i != 0 ) {

string left = "", right = "";

int startIndex, endIndex;

bool ok = false;

// Start right side;

if (expression[i + 1] == '(') {

ok = true;

for (int j = i + 2; j < expression.length(); j++) {

if (expression[j] == ')') { endIndex = j + 1; break; }

else right += expression[j];

}

}

if (!ok) {

for (int j = i + 1; j < expression.length(); j++) {

if ((expression[j] >= 48 && expression[j] <= 57) || expression[j] == '.') {

if (j == expression.length() - 1)

endIndex = j;

right += expression[j];

}

else { endIndex = j; break; }

}

}

cout << "RIGHT : " << right << '\n';

// Done right side;

// Start left side;

ok = false;

cout << "EXrpession " << expression << '\n';

if (expression[i - 1] == ')') {

ok = true;

for (int j = i - 2; j >= 0; j--) {

if (expression[j] == '(') { startIndex = j - 1; break; }

else if (j == 0) { left += expression[j]; startIndex = 0; break; }

else left += expression[j];

}

}

cout << "LEFT : " << left << '\n';

if (!ok) {

for (int j = i - 1; j >= 0; j--) {

if (j == 0 && expression[j] == '-') {

startIndex = 0;

left += expression[j];

break;

}

if ((expression[j] >= 48 && expression[j] <= 57) || expression[j] == '.') {

if (j == 0) {

startIndex = 0;

left += expression[j];

break;

}

left += expression[j];

}

else if (j == 0) { startIndex = 0; break; }

else { startIndex = j; break; }

}

}

// cout << "LEFT : " << left << '\n';

// Done left side;

reverse (left.begin(), left.end());

cout << left << " " << right << '\n';

// cout << startIndex << " " << endIndex << "\n";

double num1 = stod(left), num2 = stod(right), ans;

// cout << expression[i] << '\n';

ans = (expression[i] == '+' ? num1 + num2 : num1 - num2);

string tmp = to\_string(ans);

// cout << tmp << '\n';

if (startIndex != 0)

if (ans < 0)

tmp = '(' + tmp + ')';

if (startIndex != 0)

for (int j = 0; j <= startIndex; j++) box.push\_back(expression[j]);

for (int j = 0; j < tmp.length(); j++) box.push\_back(tmp[j]);

for (int j = endIndex; j < expression.length(); j++) box.push\_back(expression[j]);

expression = "";

for (int j = 0; j < box.size(); j++) expression += box[j];

// cout << "Now -> " << expression << '\n';

box.clear();

return calcWithoutBracket(expression);

}

}

break;

}

case 3: {

return expression.substr(1, expression.length() - 2);

break;

}

case 0: {

if (expression[0] == '-')

return '(' + expression + ')';

return expression;

break;

}

}

return "Crash";

}

inline string sanitizeUserInput (string expression) {

// sin(x), cos(x), tan(x), sqrt(x)

for (int i = 0; i < expression.length(); i++) {

if (expression[i] == 's' && expression[i + 1] == 'i' && expression[i + 2] == 'n') {

string tmp = "";

int endInd;

for (int j = i + 4; j < expression.length(); j++) {

if (expression[j] == ')') { endInd = j; break; }

tmp += expression[j];

}

cout << tmp << '\n';

double tmpNumber = stod(tmp);

tmpNumber = sin(tmpNumber);

cout << tmpNumber << '\n';

tmp = to\_string(tmpNumber);

if (i != 0)

if (tmpNumber < 0)

tmp = '(' + tmp + ')';

cout << "last tmp : " << tmp << '\n';

for (int j = 0; j < i; j++) box.push\_back(expression[j]);

for (int j = 0; j < tmp.length(); j++) box.push\_back(tmp[j]);

for (int j = endInd + 1; j < expression.length(); j++) box.push\_back(expression[j]);

expression = "";

for (int j = 0; j < box.size(); j++)

expression += box[j];

box.clear();

return expression;

}

else if (expression[i] == 'c' && expression[i + 1] == 'o' && expression[i + 2] == 's') {

string tmp = "";

int endInd;

for (int j = i + 4; j < expression.length(); j++) {

if (expression[j] == ')') { endInd = j; break; }

tmp += expression[j];

}

cout << tmp << '\n';

double tmpNumber = stod(tmp);

tmpNumber = cos(tmpNumber);

cout << tmpNumber << '\n';

tmp = to\_string(tmpNumber);

if (i != 0)

if (tmpNumber < 0)

tmp = '(' + tmp + ')';

cout << "last tmp : " << tmp << '\n';

for (int j = 0; j < i; j++) box.push\_back(expression[j]);

for (int j = 0; j < tmp.length(); j++) box.push\_back(tmp[j]);

for (int j = endInd + 1; j < expression.length(); j++) box.push\_back(expression[j]);

expression = "";

for (int j = 0; j < box.size(); j++)

expression += box[j];

box.clear();

return expression;

}

else if (expression[i] == 't' && expression[i + 1] == 'a' && expression[i + 2] == 'n') {

string tmp = "";

int endInd;

for (int j = i + 4; j < expression.length(); j++) {

if (expression[j] == ')') { endInd = j; break; }

tmp += expression[j];

}

cout << tmp << '\n';

double tmpNumber = stod(tmp);

tmpNumber = tan(tmpNumber);

cout << tmpNumber << '\n';

tmp = to\_string(tmpNumber);

if (i != 0)

if (tmpNumber < 0)

tmp = '(' + tmp + ')';

cout << "last tmp : " << tmp << '\n';

for (int j = 0; j < i; j++) box.push\_back(expression[j]);

for (int j = 0; j < tmp.length(); j++) box.push\_back(tmp[j]);

for (int j = endInd + 1; j < expression.length(); j++) box.push\_back(expression[j]);

expression = "";

for (int j = 0; j < box.size(); j++)

expression += box[j];

box.clear();

return expression;

}

else if (expression[i] == 's' && expression[i + 1] == 'q' && expression[i + 2] == 'r' && expression[i + 3] == 't') {

string tmp = "";

int endInd;

for (int j = i + 5; j < expression.length(); j++) {

if (expression[j] == ')') { endInd = j; break; }

tmp += expression[j];

}

// cout << tmp << '\n';

double tmpNumber = stod(tmp);

tmpNumber = sqrt(tmpNumber);

// cout << tmpNumber << '\n';

tmp = to\_string(tmpNumber);

if (tmpNumber < 0)

tmp = '(' + tmp + ')';

// cout << "last tmp : " << tmp << '\n';

for (int j = 0; j < i; j++) box.push\_back(expression[j]);

for (int j = 0; j < tmp.length(); j++) box.push\_back(tmp[j]);

for (int j = endInd + 1; j < expression.length(); j++) box.push\_back(expression[j]);

expression = "";

for (int j = 0; j < box.size(); j++)

expression += box[j];

box.clear();

return expression;

}

}

done = 1;

return expression;

}

inline bool isDoneBracket (string expression) {

return true;

}

inline string powerOfRecursive (string value) {

string newValue = "";

bool ok = false;

for (int i = 0; i < value.length(); i++) {

if (value[i] == '(') {

ok = true;

int endInd, startInd = i;

string tmp = "";

for (int j = i + 1; j < value.length(); j++) {

if (value[j] == ')') {

endInd = j;

break;

}

tmp += value[j];

}

tmp = calcWithoutBracket(tmp);

for (int j = 0; j < startInd; j++)

newValue += value[j];

for (int j = 0; j < tmp.length(); j++)

newValue += tmp[j];

for (int j = endInd + 1; j < value.length(); j++)

newValue += value[j];

cout << "New value : " << newValue << '\n';

if (isDoneBracket(newValue)) {

cout << "returning : " << newValue << '\n';

return newValue;

}

newValue = powerOfRecursive(newValue);

}

}

if (!ok) newValue = value;

cout << "Returning : " << newValue << '\n';

return newValue;

}

int main() {

while (true) {

pair<string,string> data;

cout << "Enter your expression : ";

string first;

cin >> first;

data.first = first;

while (true) {

cout << first << '\n';

first = sanitizeUserInput(first);

if (done == 1)

break;

}

// sqrt(36)+567-sin(23)\*572/cos(34)

// (12313\*23/123)+(42342-123+123-sin(23)+32)

// 123\*sin(223)+2323/231\*sqrt(36)-(123+323-123/23+213)

// 1234+sin(123)-(sin(32)+123\*123/23-123)/(323\*23-sqrt(64))

// (123123\*23123123/1123\*123-12313)-123123\*9213+923-12.3128/tan(23)

// sin(23)\*(2131/23)\*3-(2323-sin(34))\*sqrt(63)+23\*123-123/34\*1-234/123

cout << first << '\n';

cout << "Power of recursive starting \n";

string last = powerOfRecursive(first);

cout << "Calculator starting\n";

string finals = calcWithoutBracket(last);

// string last = powerOfRecursive(first);

cout << "Answer : " << finals << '\n';

data.second = finals;

history.push\_back(data);

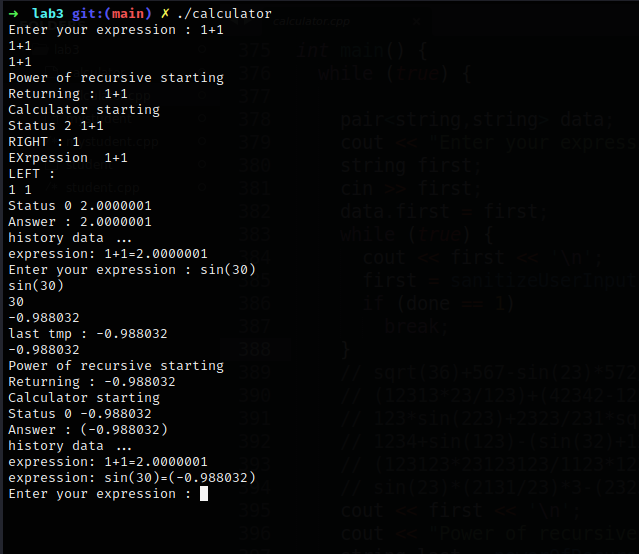
cout << "history data ..." << endl;

for (int i = 0; i < history.size(); i++) {

cout << "expression: " << history[i].first << "=" << history[i].second << endl;

}

}}



**Даалгавар-3:** Дараах програмыг бичнэ үү.

1. Оюутны дүнгийн бүртгэлийн программ. Жич: Оюутны дүнгийн бүртгэлийн програм нь:
   1. Оюутны бүртгэлийн хэсэгтэй байна. Бүртгэлийн мэдээлэлд оюутны код, овог, нэр ... гэх мэт орно.
      1. Оюутан бүртгэх
      2. Оюутны мэдээлийг засах
      3. Оюутны бүртгэлийг устгах ... гэх мэт боломжтой байна.
   2. Хичээлийн бүртгэл. Бүртгэлийн мэдээлэлд хичээлийн код, нэр, кредит, улирал ... гэх мэт орно.
      1. Хичээл бүртгэх
      2. Хичээлийн мэдээлийг засах
      3. Хичээлийн бүртгэлийг устгах ... гэх мэт боломжтой байна.
   3. Оюутны дүнгийн бүртгэл. Оюутны код, хичээлийн код хоёроор харгалзуулан бүртгэл хийнэ.
      1. Оюутны үзсэн хичээлийг бүргэх
      2. Буруу оруулсан дүнг засах
      3. Оюутны үзсэн хичээлийг устгах ... гэх мэт боломжтой байна.
   4. Тайлан, статистик гаргах
      1. Оюутнуудын жагсаалт
      2. Оюутнуудын голч дүнтэй жагсаалт
      3. Оюутны үзсэн хичээлийн жагсаалт дүнгийн хамт
   5. ... гэх мэтчилэнгээр олон үйлдэл, олон тайлан статистик гаргадаг байх тусмаа сайн.

**Жич:** Даалгаврыг гүйцэтгэхдээ заавал бүтэц ашиглана. Массив, заагч, тэмдэгт мөр... гэх мэт ойлголтууд ашиглагдах нь ойлгомжтой.

#include <iostream>

#include <vector>

#include <utility>

#include <string>

#include <map>

#include <algorithm>

#define writeLine cout << "============================================================\n"

#define localLine cout << "<==========================================================>\n"

#define successLine cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

#define endLine cout << "############################################################\n"

#define teacherLine cout << "<------------------------Багшийн веб----------------------->\n"

#define teacherLineRegister cout << "<-------------------Багшийн веб бүртгүүлэх----------------->\n"

#define teacherLineLogin cout << "<---------------------Багшийн веб нэвтрэх------------------>\n"

#define studentLine cout << "<------------------------Оюутны веб------------------------>\n"

#define studentLineRegister cout << "<-------------------Оюутны веб бүртгүүлэх------------------>\n"

#define studentLineLogin cout << "<---------------------Оюутны веб нэвтрэх------------------->\n"

#define adminLine cout << "[\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Админ веб\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*]\n"

#define pb push\_back

#define mp make\_pair

#define ff first

#define ss second

using namespace std;

typedef unsigned int ui;

typedef long long i64;

struct lesson {

string lessonName;

string lessonCode;

int lessonKr;

};

struct dataBaseTeacher {

string code;

string name;

string password;

string lastName;

vector <pair<string, pair <string, int> > > teacherStudents;

vector <lesson> lessons;

};

// Student database

struct dataBaseStudent {

string code;

string name;

string lastName;

string password;

int course;

int totalCredit;

int golch = 0;

string teacherName;

vector <pair<string, float> > studentLessons;

};

struct dataBaseAdmin {

string name;

string password;

string season = "2-A";

};

// Some global variable

vector <dataBaseTeacher> teachers;

vector <dataBaseStudent> students;

vector <dataBaseAdmin> admins;

dataBaseAdmin DefaultAdmin = {

"root", "toor"

};

dataBaseStudent DefaultValue = {

"B200970314", "Ulambayar", "Luvsanjargal", "pwd", 2, 0

};

dataBaseTeacher DefaultValueTeacher = {

"D.HW08", "Б.МӨНХБАЯР", "tmp", "Овог"

};

lesson DefaultLesson = {

"СүлжээнийПрограмчлал", "F.NS351", 3

};

string sessionStudentName = "", sessionStudentPassword = "", sessionStudentCode = "", sessionTeacherName = "", sessionTeacherPassword = "", sessionTeacherCode = "";

int sessionStudentIndex = 0, sessionTeacherIndex, sessionStudentCredit = 0, sessionTeacherCredit = 0, tmp;

inline void MainTeacherFunction(); inline void StudentAuthorized(); inline bool isInDatabaseStudent(string a); inline void TeacherAuthorized(); inline void TeacherLogin(); inline void TeacherAddGrade();

inline void MainStudentFunction(); inline bool isValidStudentCode(string outValue); inline void TeacherAddNewStudent(); inline bool isValidTeacherCode(string a); inline void TeacherAddLesson(); inline void TeacherEditGrade();

inline void MainAdminFunction(); inline int checkAvailableStudent(string a, string b); inline void TeacherShowOwnStudents(); inline bool isInDatabaseTeacher(string a); inline void TeacherPrintSalary();

inline void StudentLogin(); inline void studentPlanSchelude(); inline void studentSeeOwnInformation(); inline void TeacherMakeClassLeader(); inline void TeacherRegister(); inline float calculateStudentGolch();

inline void StudentRegister(); inline void studentSeeSchelude(); inline void studentSeeScheludesTeacher(); inline void TeacherShowOwnInformation(); inline int TeacherCheckAvailableTeacher(string a, string b);

inline void AdminFunctionAuthenticate(); inline void adminStudent(); inline void adminTeacher();

inline bool TeacherIsValidTeacherCode(string a); inline bool TeacherIsInDatabaseTeacher(string a); inline void studentsEditInformation(); inline void TeacherRemoveStudentFromCLass();

namespace GlobalMainFunction {

inline void start() {

Start:

cout << "\t\tШУТИС-МХТС вебд тавтай морил\n";

cout << "1) Оюутны веб\n2) Багшийн веб\n3) Админ\n4) Гарах\n";

cout << "Таны сонголт : ";

int choise;

cin >> choise;

if (choise != 4 && choise != 3 && choise != 2 && choise != 1) {

cout << "Таны оруулсан утгатай комманд олдсонгүй.!\n";

writeLine;

goto Start;

}

if (choise == 4) {

cout << "Манай үйлчилгээг ашигласанд баярлалаа.!\n";

endLine;

exit(0);

}

switch (choise) {

case 1: { MainStudentFunction(); break; }

case 2: { MainTeacherFunction(); break; }

case 3: { MainAdminFunction(); break; }

default : { GlobalMainFunction::start(); break; }

}

return;

}

}

inline void MainStudentFunction() {

studentLine;

cout << "1) Нэвтрэх \n2) Бүртгүүлэх\n3) Үндсэн веб\nТаны сонголт : ";

cin >> tmp;

switch (tmp) {

case 1 : { StudentLogin(); break; }

case 2 : { StudentRegister(); break; }

default : { GlobalMainFunction::start(); break; }

}

}

inline void StudentLogin() {

string studentCode, studentPassword;

int tryAgain = 1;

Start:

while (true) {

studentLineLogin;

cout << "1) Буцах\n";

cout << "Оюутны кодоо хийнэ үү : ";

cin >> studentCode;

if (studentCode == "1") { MainStudentFunction(); }

if (isValidStudentCode(studentCode)) break;

cout << "Оюутны код формат буруу байна (B170970065)\n";

}

cout << "Нууц үгээ хийнэ үү : ";

cin >> studentPassword;

int status = checkAvailableStudent(studentCode, studentPassword);

switch(status) {

case 1: {

cout << "Нууц үг буруу байна.!\n";

if (5 - tryAgain < 1) {

cout << "Уучлаарай 5 удаа нууц үг буруу байсан тул таныг системээс гаргалаа.\n";

endLine;

exit(0);

}

cout << 5 - tryAgain << " боломж үлдлээ.!\n";

tryAgain++;

goto Start;

}

case 2: {

cout << "Оюутны код системд бүртгэгдээгүй байна.\n";

goto Start;

}

case 3: {

StudentAuthorized();

break;

}

}

StudentAuthorized();

}

inline bool isValidStudentCode(string outValue) {

if (outValue[0] != 'B' || outValue.length() != 10)

return false;

return true;

}

inline void StudentRegister() {

string localName, localPassword, localCode, tmp, locallastName;

int localCredit, localCourse;

Start:

studentLineRegister;

cout << "1) Буцах\n";

cout << "Нэрээ хийнэ үү : ";

cin >> localName;

if (localName == "1") {

MainStudentFunction();

}

else if (localName.length() > 100 || localName.length() < 4) {

cout << "Нэрний урт заавал 4 өөс их 100 аас бага байх ёстой.\n";

goto Start;

}

cout << "Овгоо хийнэ үү : ";

cin >> locallastName;

cout << "Оюутны кодоо хийнэ үү : ";

cin >> localCode;

if (!isValidStudentCode(localCode)) {

cout << "Оюутны код формат буруу байна (B170970065)\n";

goto Start;

}

if (isInDatabaseStudent(localCode)) {

cout << "Дараах оюутны код манай системд бүртгэлтэй байна.!\n";

goto Start;

}

localCourse = 19 - stoi(localCode.substr(1, 3));

if (localCourse > 0) {

cout << "Суралцаж буй курс алдаатай байна.!\n";

goto Start;

}

cout << "Суралцсан нийт кредит : ";

cin >> localCredit;

if (localCredit > 200 || localCredit < 0) {

cout << "Суралцсан кредит алдаатай байна.!\n";

goto Start;

}

cout << "Бүртгүүлэх нууц үг : ";

cin >> localPassword;

cout << "Бүртгүүлэх нууц үг (баталгаажуулах) : ";

cin >> tmp;

if (localPassword != tmp) {

cout << "Нууц үг таарахгүй байна.!\n";

goto Start;

}

dataBaseStudent tmpStruct;

tmpStruct.name = localName;

tmpStruct.code = localCode;

tmpStruct.password = localPassword;

tmpStruct.totalCredit = localCredit;

tmpStruct.lastName = locallastName;

students.push\_back(tmpStruct);

successLine;

cout << "Бүртгэл амжилттай боллоо.!\n\n";

MainStudentFunction();

}

inline int checkAvailableStudent(string studentCode, string studentPassword) {

for (ui it = 0; it < students.size(); ++it) {

if (students[it].code == studentCode && students[it].password == studentPassword) {

sessionStudentName = students[it].name;

sessionStudentPassword = students[it].password;

sessionStudentCode = students[it].code;

sessionStudentIndex = it;

sessionStudentCredit = students[it].totalCredit;

return 3;

}

else if (students[it].code == studentCode && students[it].password != studentPassword) {

return 1;

}

}

return 2;

}

inline void StudentAuthorized() {

if (sessionStudentName == "" || sessionStudentIndex == -1)

MainStudentFunction();

cout << "\n\t\tТавтай морил # " << sessionStudentName << '\n';

successLine;

int tmpReq;

Start:

float golch = calculateStudentGolch();

students[sessionStudentIndex].golch = golch;

cout << "1) Хичээл сонгох\n2) Сонгосон хичээлээ харах\n";

cout << "3) Хувийн мэдээлэл\n";

cout << "4) Сонгосон багшаа харах\n5) Нууц үг өөрчлөх\n6) Гарах\n";

cout << "Таны сонголт : ";

cin >> tmpReq;

switch (tmpReq) {

case 1: { localLine; studentPlanSchelude(); break; }

case 2: { localLine; studentSeeSchelude(); break; }

case 3: { localLine; studentSeeOwnInformation(); break; }

case 4: { localLine; studentSeeScheludesTeacher(); break; }

case 5: { localLine; studentsEditInformation(); break; }

case 6: {

sessionStudentName = "";

sessionStudentPassword = "";

sessionStudentCode = "";

sessionStudentIndex = 0;

sessionStudentCredit = 0;

MainStudentFunction();

}

default: {

cout << "Таны сонгосон үйлдэл одоогоор боломжгүй байна.!\n";

localLine;

goto Start;

}

}

}

inline void studentsEditInformation() {

cout << "1) Буцах \n###Оюуны нэр, кодыг зөвхөн админ л засах боломжтой.!###\n";

string tmpPassword, newPassword, new2Password;

cout << "Хуучин нууц үг : "; cin >> tmpPassword;

if (tmpPassword == "1") StudentAuthorized();

cout << "Шинэ нууц үг : "; cin >> newPassword;

cout << "Шинэ нууц үг (баталгаажуулах): "; cin >> new2Password;

if (sessionStudentPassword != tmpPassword) {

cout << "Хуучин нууц үг буруу байна.!\n";

studentsEditInformation();

}

if (newPassword != new2Password) {

cout << "Нууц үг таарахгүй байна.\n";

studentsEditInformation();

}

cout << "Нууц үг амжилттай өөрчиллөө.!\n";

students[sessionStudentIndex].password = newPassword;

sessionStudentName = "";

sessionTeacherIndex = -1;

StudentAuthorized();

}

inline bool isInDatabaseStudent (string outCode) {

for (ui it = 0; it < students.size(); ++it)

if (students[it].code == outCode)

return true;

return false;

}

inline void studentPlanSchelude() {

string tmpLesson;

int count = 0;

cout << "Идэвхитэй байгаа хичээлүүд : \n";

for (ui i = 0; i < teachers.size(); i++) {

for (ui j = 0; j < teachers[i].lessons.size(); j++) {

cout << "\tХичээлийн нэр : " << teachers[i].lessons[j].lessonName << '\n' << "\tКредит : " << teachers[i].lessons[j].lessonKr << "\n\tХичээлийн код : " << teachers[i].lessons[j].lessonCode << '\n';

count++;

localLine;

}

}

if (count == 0) {

cout << "Одоогийн байдлаар ямар ч багш хичээл оруулаагүй байна.!\n";

StudentAuthorized();

}

cout << "Сонгох хичээлийн нэрийг оруулна уу : ";

cin >> tmpLesson;

bool flag = false;

for (ui i = 0; i < teachers.size(); i++) {

for (ui j = 0; j < teachers[i].lessons.size(); j++) {

if (teachers[i].lessons[j].lessonName == tmpLesson) {

students[sessionStudentIndex].studentLessons.push\_back(make\_pair(tmpLesson, 0));

flag = true;

break;

}

}

}

if (!flag) {

cout << "Таны сонгосон хичээл олдсонгүй.!\n";

}

if (flag) { cout << "Амжилттай хичээл сонголоо.!\n"; StudentAuthorized(); }

return;

}

inline void studentSeeSchelude() {

if (students[sessionStudentIndex].studentLessons.size() == 0) {

cout << "Та одоогоор ямар нэгэн хичээл сонгоогүй байна.\n";

StudentAuthorized();

}

for (int i = 0; i < students[sessionStudentIndex].studentLessons.size(); i++) {

cout << '\t' << students[sessionStudentIndex].studentLessons[i].first << '\n';

}

StudentAuthorized();

return;

}

inline void studentSeeOwnInformation() {

cout << "Оюутны код : " << sessionStudentCode << '\n';

cout << "Оюутны овог : " << students[sessionStudentIndex].lastName << '\n';

cout << "Оюутны нэр : " << sessionStudentName << '\n';

cout << "Суралцах буй курс : " << students[sessionStudentIndex].course << '\n';

cout << "Нийт судалсан кредит : " << sessionStudentCredit << '\n';

cout << "Анги даасан багшийн нэр : " << (students[sessionStudentIndex].teacherName != "" ? students[sessionStudentIndex].teacherName : "Одоохондоо тодорхойгүй") << '\n';

cout << "Улирал : " << "Хавар\n";

float golch = calculateStudentGolch();

students[sessionStudentIndex].golch = golch;

cout << "Голч : " << golch << '\n';

StudentAuthorized();

}

inline float calculateStudentGolch() {

float res = 0, count = 0;

for (int i = 0 ; i < students[sessionStudentIndex].studentLessons.size(); i++) {

res += students[sessionStudentIndex].studentLessons[i].second;

count++;

}

return (res / count);

}

inline void studentSeeScheludesTeacher() {

int totalKre = 0;

for (int i = 0; i < students[sessionStudentIndex].studentLessons.size(); ++i) {

for (int j = 0; j < teachers.size(); j++) {

for (int k = 0; k < teachers[j].teacherStudents.size(); k++) {

if (students[sessionStudentIndex].studentLessons[i].first == teachers[i].lessons[j].lessonName) {

cout << "Хичээлийн нэр : " << students[sessionStudentIndex].studentLessons[i].first << '\n';

cout << "Багшийн нэр : " << teachers[i].name << '\n';

cout << "Кредит : " << teachers[i].lessons[j].lessonKr;

writeLine;

totalKre += teachers[i].lessons[j].lessonKr;

}

}

}

}

cout << "Энэ улирал " << totalKre << " кредит судалж байна.\n";

StudentAuthorized();

return;

}

inline void MainTeacherFunction() {

teacherLine;

cout << "1) Нэвтрэх \n2) Бүртгүүлэх\n3) Буцах\nТаны сонголт : ";

int tmp; cin >> tmp;

switch (tmp) {

case 1: { TeacherLogin(); break; } case 2: { TeacherRegister(); break; }

default: { GlobalMainFunction::start(); }

}

}

inline void Register() {

string localName, localPassword, localCode, locallastName, tmp;

int localCredit, localCourse;

Start:

teacherLineRegister;

cout << "1) Буцах\n";

cout << "Нэрээ хийнэ үү : ";

cin >> localName;

if (localName == "1")

MainTeacherFunction();

else if (localName.length() > 100 || localName.length() < 4) {

cout << "Нэрний урт заавал 4 өөс их 100 аас бага байх ёстой.\n";

goto Start;

}

cout << "Овогоо хийнэ үү : ";

cin >> locallastName;

cout << "Багшийн кодоо хийнэ үү : ";

cin >> localCode;

if (!TeacherIsValidTeacherCode(localCode)) {

cout << "Багшийн код формат буруу байна (D.XY12)\n";

goto Start;

}

if (TeacherIsInDatabaseTeacher(localCode)) {

cout << "Дараах багшийн код манай системд бүртгэлтэй байна.!\n";

goto Start;

}

cout << "Бүртгүүлэх нууц үг : ";

cin >> localPassword;

cout << "Бүртгүүлэх нууц үг (баталгаажуулах) : ";

cin >> tmp;

if (localPassword != tmp) {

cout << "Нууц үг таарахгүй байна.!\n";

goto Start;

}

dataBaseTeacher tmpStruct;

tmpStruct.name = localName;

tmpStruct.code = localCode;

tmpStruct.lastName = locallastName;

tmpStruct.password = localPassword;

teachers.push\_back(tmpStruct);

successLine;

cout << "Бүртгэл амжилттай боллоо.!\n\n";

MainTeacherFunction();

}

inline void TeacherAuthorized() {

successLine;

cout << "\n\t\tТавтай морил # " << sessionTeacherName << '\n';

int tmpReq;

Start:

teacherLine;

cout << "1) Ангидаа шинэ хүүхэд нэмэх\n2) Ангийн сурагчдаа харах\n";

cout << "3) Ангийн дарга томилох\n";

cout << "4) Хувийн мэдээлэл харах\n5) Хичээл нэмэх\n6) Ангиас хүүхэд хасах\n7) Дүн оруулах\n8) Цалин бодох\n9) Гарах\n";

cout << "Таны сонголт : ";

cin >> tmpReq;

switch (tmpReq) {

case 1: { localLine; TeacherAddNewStudent(); break; }

case 2: { localLine; TeacherShowOwnStudents(); break; }

case 3: { localLine; TeacherMakeClassLeader(); break; }

case 4: { localLine; TeacherShowOwnInformation(); break; }

case 5: { localLine; TeacherAddLesson(); break; }

case 6: { localLine; TeacherRemoveStudentFromCLass(); break; }

case 7: { localLine; TeacherAddGrade(); break; }

case 8: { localLine; TeacherPrintSalary(); break; }

case 9: {

sessionTeacherName = "";

sessionTeacherPassword = "";

sessionTeacherCode = "";

sessionTeacherIndex = 0;

MainTeacherFunction(); break;

}

default: { MainTeacherFunction(); break; }

}

}

inline void TeacherAddGrade() {

string tmpName; int tmpIndex;

cout << "Таны энэ улирал зааж буй хичээлүүд : \n";

int count = 0, kr;

for (int ui = 0; ui < teachers[sessionTeacherIndex].lessons.size(); ui++) {

cout << "\t[\*] " << teachers[sessionTeacherIndex].lessons[ui].lessonName << '\n';

count++;

}

if (count == 0) {

cout << "Танд энэ улиралд зааж буй хичээл алга.!\n";

TeacherAuthorized();

}

cout << "Дүн оруулах хичээлийн нэр : ";

cin >> tmpName;

for (int ui = 0; ui < teachers[sessionTeacherIndex].lessons.size(); ui++) if (teachers[sessionTeacherIndex].lessons[ui].lessonName == tmpName) {

tmpIndex = ui;

kr = teachers[sessionTeacherIndex].lessons[ui].lessonKr;

break;

}

cout << "Тухайн хичээл дээр бүртгэлтэй байгаа оюутнууд харгалзан дүнг оруулна уу : \n";

for (int i = 0; i < students.size(); i++) {

for (int j = 0; j < students[i].studentLessons.size(); j++) {

if (students[i].studentLessons[j].first == tmpName) {

cout << "\t[\*] " << students[i].name << " -> ";

float grade; cin >> grade; if (grade > 4.0 || grade < 0.0) { cout << "Дүнгийн мэдээлэл алдаатай байна.!"; TeacherAddGrade(); }

students[i].studentLessons[j].second = grade;

students[i].totalCredit += kr;

}

}

}

TeacherAuthorized();

}

inline void TeacherPrintSalary() {

string tmpName; int tmpIndex;

int count = 0;

for (int ui = 0; ui < teachers[sessionTeacherIndex].lessons.size(); ui++) {

cout << "\t[\*] " << teachers[sessionTeacherIndex].lessons[ui].lessonName << '\n';

count += teachers[sessionTeacherIndex].lessons[ui].lessonKr;

}

cout << "\nТаны энэ улирал авах цалин : " << count \* 78900 << '\n';

TeacherAuthorized();

}

inline void TeacherAddLesson() {

string tmpName, tmpCode;

int kr;

cout << "Хичээлийн нэр : "; cin >> tmpName;

cout << "Хичээлийн код : "; cin >> tmpCode;

cout << "Кредит : "; cin >> kr;

lesson tmpLesson;

tmpLesson.lessonName = tmpName;

tmpLesson.lessonCode = tmpCode;

tmpLesson.lessonKr = kr;

teachers[sessionTeacherIndex].lessons.push\_back(tmpLesson);

cout << "Хичээл амжилттай нэмлээ.!\n";

TeacherAuthorized();

}

inline void TeacherRemoveStudentFromCLass() {

if (teachers[sessionTeacherIndex].teacherStudents.size() == 0) {

cout << "Тухайн багш дээр бүртгэлтэй оюутан олдсонгүй.!\n";

TeacherAuthorized();

}

sort(teachers[sessionTeacherIndex].teacherStudents.begin(), teachers[sessionTeacherIndex].teacherStudents.end());

cout << sessionTeacherName << " багш дээр бүртгэлтэй оюутнууд : \n";

for (unsigned int it = 0; it < teachers[sessionTeacherIndex].teacherStudents.size(); ++it) {

if (teachers[sessionTeacherIndex].teacherStudents[it].second.second == 1)

cout << "[\*] " << teachers[sessionTeacherIndex].teacherStudents[it].first << " " << teachers[sessionTeacherIndex].teacherStudents[it].second.first << " (ангийн дарга)\n";

else

cout << "[\*] " << teachers[sessionTeacherIndex].teacherStudents[it].first << " " << teachers[sessionTeacherIndex].teacherStudents[it].second.first << '\n';

} string tmpUserName;

cout << "Өөрийн ангиас хасах хүүхдийн нэр : "; cin >> tmpUserName;

for (unsigned int it = 0; it < teachers[sessionTeacherIndex].teacherStudents.size(); ++it) if (teachers[sessionTeacherIndex].teacherStudents[it].second.first == tmpUserName) {

teachers[sessionTeacherIndex].teacherStudents.erase(teachers[sessionTeacherIndex].teacherStudents.begin() + it);

cout << "Амжилттай хаслаа.!\n";

TeacherAuthorized();

break;

}

cout << "Тухайн оюутныг олсонгүй.\n";

TeacherAuthorized();

}

inline void TeacherAddNewStudent() {

Start:

int tmpCount = 0; string tmpStudentName; bool fine = false;

cout << "Ямар нэгэн ангид бүртгэлгүй байгаа оюутнууд : \n";

for (ui i = 0; i < students.size(); ++i) if (students[i].teacherName == "") {

cout << "[\*] " << students[i].code << " " << students[i].name << '\n'; tmpCount++;

}

localLine;

if (tmpCount == 0) { cout << "Бүх оюутан багштай эсвэл системд ямар нэгэн оюутан бүртгэгдээгүй байна.!\n"; TeacherAuthorized(); }

cout << "1) Буцах\nАнгидаа нэмэх оюутны нэр эсвэл оюутны код : "; cin >> tmpStudentName; if (tmpStudentName == "1") TeacherAuthorized();

for (ui i = 0; i < students.size(); i++) if (students[i].name == tmpStudentName || students[i].code == tmpStudentName) {

teachers[sessionTeacherIndex].teacherStudents.push\_back(make\_pair(students[i].code, make\_pair(students[i].name, 0)));

students[i].teacherName = teachers[sessionTeacherIndex].name;

cout << "Тус оюутныг өөрийн ангидаа амжилттай нэмлээ.!\n";

TeacherAuthorized();

fine = true;

}

if (!fine) cout << "Оюутанг олсонгүй.!\n";

goto Start;

}

inline void TeacherShowOwnStudents() {

Start:

if (teachers[sessionTeacherIndex].teacherStudents.size() == 0) {

cout << "Тухайн багш дээр бүртгэлтэй оюутан олдсонгүй.!\n";

TeacherAuthorized();

}

sort(teachers[sessionTeacherIndex].teacherStudents.begin(), teachers[sessionTeacherIndex].teacherStudents.end());

cout << sessionTeacherName << " багш дээр бүртгэлтэй оюутнууд : \n";

for (unsigned int it = 0; it < teachers[sessionTeacherIndex].teacherStudents.size(); ++it) {

if (teachers[sessionTeacherIndex].teacherStudents[it].second.second == 1)

cout << "[\*] " << teachers[sessionTeacherIndex].teacherStudents[it].first << " " << teachers[sessionTeacherIndex].teacherStudents[it].second.first << " (ангийн дарга)\n";

else

cout << "[\*] " << teachers[sessionTeacherIndex].teacherStudents[it].first << " " << teachers[sessionTeacherIndex].teacherStudents[it].second.first << '\n';

}

TeacherAuthorized();

}

inline void TeacherMakeClassLeader() {

Start:

int tmpIndex = -1, count = 0; string tmpStudentName;

vector <pair <string, int > > tmpVector;

if (teachers[sessionTeacherIndex].teacherStudents.size() == 0) {

cout << "Тухайн багш дээр бүртгэлтэй оюутан олдсонгүй.!\n";

TeacherAuthorized();

}

sort(teachers[sessionTeacherIndex].teacherStudents.begin(), teachers[sessionTeacherIndex].teacherStudents.end());

cout << sessionTeacherName << " багш дээр бүртгэлтэй оюутнууд : \n";

for (ui it = 0; it < teachers[sessionTeacherIndex].teacherStudents.size(); ++it) {

tmpVector.push\_back(make\_pair(teachers[sessionTeacherIndex].teacherStudents[it].second.first, it));

if (teachers[sessionTeacherIndex].teacherStudents[it].second.second == 1) {

cout << "[\*] " << teachers[sessionTeacherIndex].teacherStudents[it].second.first << " (ангийн дарга)\n";

count++;

}

else

cout << "[\*] " << teachers[sessionTeacherIndex].teacherStudents[it].second.first << '\n';

}

cout << (count > 0 ? "Нэмж ангийн дарга болгох оюутны нэр : " : "Ангийн дарга болгох оюутны нэр : ");

cin >> tmpStudentName;

for (ui i = 0; i < tmpVector.size(); ++i) if (tmpVector[i].first == tmpStudentName) {

tmpIndex = i; break;

}

if (tmpIndex == -1) { cout << "Оюутан олдсонгүй\n"; TeacherAuthorized(); }

teachers[sessionTeacherIndex].teacherStudents[tmpIndex].second.second = 1;

cout << "Амжилттай ангийн дарга томиллоо.!\n";

TeacherAuthorized();

}

inline void TeacherShowOwnInformation() {

cout << "Багшийн код : " << sessionTeacherCode << '\n';

cout << "Багшийн овог : " << teachers[sessionTeacherIndex].lastName << '\n';

cout << "Багшийн нэр : " << sessionTeacherName << '\n';

cout << "Улирал : " << "Хавар\n";

TeacherAuthorized();

}

inline void TeacherLogin() {

string teacherCode, teacherPassword;

int tryAgain = 1;

Start:

while (true) {

teacherLineLogin;

cout << "1) Буцах\n";

cout << "Багшийн кодоо хийнэ үү : ";

cin >> teacherCode;

if (teacherCode == "1") { MainTeacherFunction(); }

if (TeacherIsValidTeacherCode(teacherCode)) break;

cout << "Багшийн код формат буруу байна (D.XY12)\n";

}

cout << "Нууц үгээ хийнэ үү : ";

cin >> teacherPassword;

int status = TeacherCheckAvailableTeacher(teacherCode, teacherPassword);

switch (status) {

case 1: {

cout << "Нууц үг буруу байна.!\n";

if (5 - tryAgain < 1) {

cout << "Уучлаарай 5 удаа нууц үг буруу байсан тул таныг системээс гаргалаа.\n";

endLine;

exit(0);

}

cout << 5 - tryAgain << " боломж үлдлээ.!\n";

tryAgain++;

goto Start;

}

case 2: {

cout << "Багшийн код системд бүртгэгдээгүй байна.\n";

goto Start;

}

case 3: {

TeacherAuthorized();

break;

}

}

TeacherAuthorized();

}

inline void TeacherRegister() {

string localName, localPassword, localCode, localLastName, tmp;

int localCredit, localCourse;

Start:

teacherLineRegister;

cout << "1) Буцах\n";

cout << "Нэрээ хийнэ үү : ";

cin >> localName;

if (localName == "1")

MainTeacherFunction();

else if (localName.length() > 100 || localName.length() < 4) {

cout << "Нэрний урт заавал 4 өөс их 100 аас бага байх ёстой.\n";

goto Start;

}

cout << "Овгоо хийнэ үү : ";

cin >> localLastName;

cout << "Багшийн кодоо хийнэ үү : ";

cin >> localCode;

if (!TeacherIsValidTeacherCode(localCode)) {

cout << "Багшийн код формат буруу байна (D.XY12)\n";

goto Start;

}

if (TeacherIsInDatabaseTeacher(localCode)) {

cout << "Дараах багшийн код манай системд бүртгэлтэй байна.!\n";

goto Start;

}

cout << "Бүртгүүлэх нууц үг : ";

cin >> localPassword;

cout << "Бүртгүүлэх нууц үг (баталгаажуулах) : ";

cin >> tmp;

if (localPassword != tmp) {

cout << "Нууц үг таарахгүй байна.!\n";

goto Start;

}

dataBaseTeacher tmpStruct;

tmpStruct.name = localName;

tmpStruct.code = localCode;

tmpStruct.lastName = localLastName;

tmpStruct.password = localPassword;

teachers.push\_back(tmpStruct);

successLine;

cout << "Бүртгэл амжилттай боллоо.!\n\n";

MainTeacherFunction();

}

inline int TeacherCheckAvailableTeacher(string teacherCode, string teacherPassword) {

for (ui it = 0; it < teachers.size(); ++it) {

if (teachers[it].code == teacherCode && teachers[it].password == teacherPassword) {

sessionTeacherName = teachers[it].name;

sessionTeacherPassword = teachers[it].password;

sessionTeacherCode = teachers[it].code;

sessionTeacherIndex = it;

return 3;

}

else if (teachers[it].code == teacherCode && teachers[it].password != teacherPassword) {

return 1;

}

}

return 2;

}

inline bool TeacherIsInDatabaseTeacher (string outCode) {

for (ui it = 0; it < teachers.size(); ++it)

if (teachers[it].code == outCode)

return true;

return false;

}

inline bool TeacherIsValidTeacherCode(string outValue) {

if (outValue.length() != 6)

return false;

return true;

}

inline void MainAdminFunction() {

string tmpUsername, tmpUserpassword;

cout << "Username : "; cin >> tmpUsername;

cout << "Password : "; cin >> tmpUserpassword;

if (admins[0].name != tmpUsername || admins[0].password != tmpUserpassword) {

cout << "Authenticantin failed\n";

GlobalMainFunction::start();

}

AdminFunctionAuthenticate();

return;

}

inline void AdminFunctionAuthenticate() {

cout << "1) Оюутнуудын мэдээлэл харах\n2) Багш нарын мэдээлэл харах\n3) Буцах\nТаны сонголт : ";

int req; cin >> req;

switch (req) {

case 1: { adminStudent(); break; }

case 2: { adminTeacher(); break; }

default : {GlobalMainFunction::start(); break; }

}

}

inline void adminStudent() {

cout << "Бүх оюутнуудын дүнгийн жагсаалт : \n";

for (int i = 0; i < students.size(); i++) {

float totalGolch = 0; float count = 0;

for (int j = 0; j < students[i].studentLessons.size(); j++) {

totalGolch += students[i].studentLessons[j].second;

count++;

}

cout << "\t[\*] " << students[i].name << " " << totalGolch / count << '\n';

}

AdminFunctionAuthenticate();

}

inline void adminTeacher() {

cout << "Бүх багш нарын мэдээлэл : \n";

for (int i = 0; i < teachers.size(); i++) {

int k = 0;

cout << "\tБагшийн нэр : " << teachers[i].name << '\n';

cout << "\tБагшийн овог: " << teachers[i].lastName << '\n';

cout << "\tБагшийн код : " << teachers[i].code << '\n';

cout << "\tБагшийн нууц үг : " << teachers[i].password << '\n';

cout << "\tТухайн багшийн ангийн хүүхдүүд : \n";

for (int j = 0; j < teachers[i].teacherStudents.size(); j++) {

cout << "\t\t[\*] " << teachers[i].teacherStudents[j].first;

if (teachers[i].teacherStudents[j].second.second == 1) cout << " (ангийн дарга)\n";

cout << '\n';

}

cout << "\tТухайн багшийн энэ улирал зааж буй хичээлүүд : \n";

for (int j = 0; j < teachers[i].lessons.size(); j++) {

cout << "\t\t[\*] " << teachers[i].lessons[j].lessonName << '\n';

k += teachers[i].lessons[j].lessonKr;

}

cout << "\tЭнэ улиралд нийт " << k << " кредит хичээл зааж байна.!\n";

localLine;

}

AdminFunctionAuthenticate();

return;

}

int main() {

students.push\_back(DefaultValue);

teachers.push\_back(DefaultValueTeacher);

admins.push\_back(DefaultAdmin);

teachers[0].lessons.push\_back(DefaultLesson);

GlobalMainFunction::start();

return 0;

}