[17:40, 10/07/2024] David Bekham: #include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <algorithm>

struct Student {

std::string firstName;

std::string surname;

std::string gender;

int age;

int group;

std::string sport;

std::vector<std::string> clubs;

};

struct Activity {

std::string name;

int maxCapacity;

int currentCapacity;

int maleCount;

int femaleCount;

};

std::vector<Student> students;

std::vector<Activity> sports = {{"Rugby", 20, 0, 0, 0}, {"Athletics", 20, 0, 0, 0}, {"Swimming", 20, 0, 0, 0}, {"Soccer", 20, 0, 0, 0}};

std::vector<Activity> clubs = {{"Journalism Club", 60, 0, 0, 0}, {"Red Cross Society", 60, 0, 0, 0}, {"AISEC", 60, 0, 0, 0}, {"Business Club", 60, 0, 0, 0}, {"Computer Science Club", 60, 0, 0, 0}};

void addStudent();

void viewStudents();

void viewClubs();

void viewSports();

void viewGroupedStudents();

void saveAllFiles();

bool canJoinActivity(Activity& activity, const std::string& gender, bool isSport);

int main() {

int choice;

do {

std::cout << "Menu:\n";

std::cout << "1. Add Student\n";

std::cout << "2. View Students (ALL and per group)\n";

std::cout << "3. View Clubs/ Societies\n";

std::cout << "4. View Sports\n";

std::cout << "5. View Grouped Students\n";

std::cout << "6. Save all Files\n";

std::cout << "7. Exit\n";

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

addStudent();

break;

case 2:

viewStudents();

break;

case 3:

viewClubs();

break;

case 4:

viewSports();

break;

case 5:

viewGroupedStudents();

break;

case 6:

saveAllFiles();

break;

case 7:

std::cout << "Exiting...\n";

break;

default:

std::cout << "Invalid choice! Please try again.\n";

}

} while (choice != 7);

return 0;

}

void addStudent() {

Student student;

std::string sportChoice;

std::string clubChoice;

int clubCount;

std::cout << "Enter first name: ";

std::cin >> student.firstName;

std::cout << "Enter surname: ";

std::cin >> student.surname;

std::cout << "Enter gender (Male/Female): ";

std::cin >> student.gender;

std::cout << "Enter age: ";

std::cin >> student.age;

std::cout << "Enter BBIT group (1, 2, or 3): ";

std::cin >> student.group;

std::cout << "Do you want to participate in a sport? (yes/no): ";

std::cin >> sportChoice;

if (sportChoice == "yes") {

std::cout << "Available sports:\n";

for (const auto& sport : sports) {

std::cout << sport.name << " (Capacity: " << sport.currentCapacity << "/" << sport.maxCapacity << ")\n";

}

std::cout << "Enter the sport you want to join: ";

std::cin >> student.sport;

auto it = std::find\_if(sports.begin(), sports.end(), [&](const Activity& sport) { return sport.name == student.sport; });

if (it != sports.end() && canJoinActivity(\*it, student.gender, true)) {

it->currentCapacity++;

if (student.gender == "Male") {

it->maleCount++;

} else {

it->femaleCount++;

}

} else {

std::cout << "Cannot join the sport due to capacity or gender limit.\n";

student.sport = "";

}

std::cout << "How many clubs do you want to join? (0, 1, or 2): ";

std::cin >> clubCount;

for (int i = 0; i < clubCount; ++i) {

std::cout << "Available clubs:\n";

for (const auto& club : clubs) {

std::cout << club.name << " (Capacity: " << club.currentCapacity << "/" << club.maxCapacity << ")\n";

}

std::cout << "Enter the club you want to join: ";

std::cin >> clubChoice;

auto it = std::find\_if(clubs.begin(), clubs.end(), [&](const Activity& club) { return club.name == clubChoice; });

if (it != clubs.end() && canJoinActivity(\*it, student.gender, false)) {

it->currentCapacity++;

if (student.gender == "Male") {

it->maleCount++;

} else {

it->femaleCount++;

}

student.clubs.push\_back(clubChoice);

} else {

std::cout << "Cannot join the club due to capacity or gender limit.\n";

}

}

} else {

std::cout << "How many clubs do you want to join? (1 to 3): ";

std::cin >> clubCount;

for (int i = 0; i < clubCount; ++i) {

std::cout << "Available clubs:\n";

for (const auto& club : clubs) {

std::cout << club.name << " (Capacity: " << club.currentCapacity << "/" << club.maxCapacity << ")\n";

}

std::cout << "Enter the club you want to join: ";

std::cin >> clubChoice;

auto it = std::find\_if(clubs.begin(), clubs.end(), [&](const Activity& club) { return club.name == clubChoice; });

if (it != clubs.end() && canJoinActivity(\*it, student.gender, false)) {

it->currentCapacity++;

if (student.gender == "Male") {

it->maleCount++;

} else {

it->femaleCount++;

}

student.clubs.push\_back(clubChoice);

} else {

std::cout << "Cannot join the club due to capacity or gender limit.\n";

}

}

}

students.push\_back(student);

}

void viewStudents() {

for (const auto& student : students) {

std::cout << "Name: " << student.firstName << " " << student.surname << ", Gender: " << student.gender << ", Age: " << student.age << ", Group: " << student.group << "\n";

if (!student.sport.empty()) {

std::cout << "Sport: " << student.sport << "\n";

}

if (!student.clubs.empty()) {

std::cout << "Clubs: ";

for (const auto& club : student.clubs) {

std::cout << club << " ";

}

std::cout << "\n";

}

}

}

void viewClubs() {

for (const auto& club : clubs) {

std::cout << "Club: " << club.name << ", Capacity: " << club.currentCapacity << "/" << club.maxCapacity << "\n";

}

}

void viewSports() {

for (const auto& sport : sports) {

std::cout << "Sport: " << sport.name << ", Capacity: " << sport.currentCapacity << "/" << sport.maxCapacity << "\n";

}

}

void viewGroupedStudents() {

for (int group = 1; group <= 3; ++group) {

std::cout << "Group " << group << " students:\n";

for (const auto& student : students) {

if (student.group == group) {

std::cout << "Name: " << student.firstName << " " << student.surname << ", Gender: " << student.gender << ", Age: " << student.age << "\n";

if (!student.sport.empty()) {

std::cout << "Sport: " << student.sport << "\n";

}

if (!student.clubs.empty()) {

std::cout << "Clubs: ";

for (const auto& club : student.clubs) {

std::cout << club << " ";

}

std::cout << "\n";

}

}

}

}

}

void saveAllFiles() {

std::ofstream studentFile("students.csv");

studentFile << "FirstName,Surname,Gender,Age,Group,Sport,Clubs\n";

for (const auto& student : students) {

studentFile << student.firstName << "," << student.surname << "," << student.gender << "," << student.age << "," << student.group << "," << student.sport << ",";

for (size\_t i = 0; i < student.clubs.size(); ++i) {

studentFile << student.clubs[i];

if (i < student.clubs.size() - 1) {

studentFile << ";";

}

}

studentFile << "\n";

}

studentFile.close();

std::cout << "Student data saved to students.csv\n";

}

bool canJoinActivity(Activity& activity, const std::string& gender, bool isSport) {

if (activity.currentCapacity >= activity.maxCapacity) {

return false;

}

if (gender == "Male" && activity.maleCount >= activity.maxCapacity \* 0.5) {

return false;

}

if (gender == "Female" && activity.femaleCount >= activity.maxCapacity \* 0.5) {

return false;

}

return 0;

}

[17:49, 10/07/2024] David Bekham: ok guys will each paste the part will present let me reassign the roles

me-from the beginning to the struct activity

Rodgers-from vector<student> to bool can join activity

mose-intmain() to return 0;

lysa-void addstudent() to student.pushback

tendo-void viewstudents to std::cout<<\n

Then ill do the rest

[17:50, 10/07/2024] David Bekham: plz master your roles kindly and have some knowledge on what you'll present

[17:54, 10/07/2024] David Bekham: we meet tomorrow by 8am at STC 2nd floor to ready ourselves