Practice 1 - Report / Практична робота 1 - Звіт

Завдання:

Практична робота 1 **05.09.2023**

Повторити тему «**Опрацювання структур**»

Описати структуру *Футболіст* з полями :

Прізвище
Амплуа (тип перечислення)
Вік
Кількість ігор
Кількість забитих голів

Створити динамічний масив структур, записати його в бінарний файл.

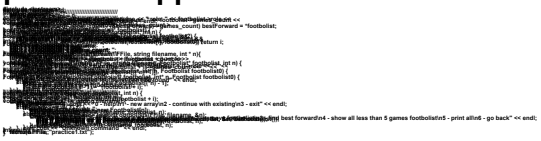
Зчитати інформацію з файлу і реалізувати функції:

визначити кращого форварда і вивести інформацію про нього;
вивести відомості про футболістів, які зіграли менше 5-ти ігор.

Реалізувати можливість додавати нових гравців і видаляти тих, хто пішов з команди.

Скрипт:

practice1.cpp:



Пояснення:

```cpp
#include <iostream>
#include <fstream>
using namespace std;



////////////////////////////////////////////////////////////////////////
struct Footbolist {
    string surname = "";
    string role = "";
    short age = 0;
    short games_count = 0;
    int goal_count = 0;
};
////////////////////////////////////////////////////////////////////////
```

Бібліотеки вводу виводу в консоль і файл, простір імен, запис структури Footbolist.

```
//////////////////////////////////////////////////////////////////
void PrintFootbolist(Footbolist* footbolist) {
    cout << "surname: " << footbolist->surname << " role: " << footbolist->role <<
        " age: " << footbolist->age << " games count: " << footbolist->games_count <<
        " goal count: " << footbolist->goal_count << endl;
}
//////////////////////////////////////////////////////////////////
```

Функція, що виводить інформацію про даного футболіста у консоль.

```
//////////////////////////////////////////////////////////////////
void BestForward(Footbolist* footbolist, int n) {
    Footbolist bestForward = *footbolist;
    footbolist++;
    while (n != 0) {
        if (footbolist->goal_count / footbolist->games_count >
            (&bestForward)->goal_count / (&bestForward)->games_count) bestForward = *footbolist;
        n--;
        footbolist++;
    }
    PrintFootbolist(&bestForward);
}
```

Функція, яка шукає в масиві футболістів з кількістю n найкращого форварда, і виводить його в консоль. Найкращим форвардом в моєму уявлені вважається футболіст, який у середньому за весь час забив найбільшу кількість голів (незалежно від головної ролі футболіста). Складність виконання алгоритму O(n).

```
bool isLessThanFiveGames(Footbolist* footbolist) {
    return footbolist->games_count < 5;
}
void LessThanFiveGames(Footbolist* footbolist, int n) {
    for (int i = 0; i < n; i++)
        if (isLessThanFiveGames(footbolist + i))
            PrintFootbolist(footbolist + i);
}
//////////////////////////////////////////////////////////////////
```

Перша функція повертає значення 1, якщо даний футболіст зіграв менше ніж 5 разів, або повертає 0, якщо це не вірно. Складність виконання алгоритму O(1).
Друга функція шукає в масиві футболістів з кількістю n усіх футболістів, які зіграли менше ніж 5 разів, і виводить їх в консоль. Складність виконання алгоритму O(n).

```
//////////////////////////////////////////////////////////////////
bool SameFootbolist(Footbolist footbolist1, Footbolist footbolist2) {
    return (&footbolist1)->surname == (&footbolist2)->surname &&
        (&footbolist1)->role == (&footbolist2)->role &&
        (&footbolist1)->age == (&footbolist2)->age &&
        (&footbolist1)->games_count == (&footbolist2)->games_count &&
        (&footbolist1)->goal_count == (&footbolist2)->goal_count;
}
int FindFootbolist(Footbolist* footbolist, int n, Footbolist footbolist0) {
    for (int i = 0; i < n; i++) if (SameFootbolist(footbolist[i], footbolist0)) return i;
    return -1;
}
```

Перша функція порівнює двух даних футболістів. Якщо футболісти однакові (всі дані збігаються), то функція повертає 1, якщо ні, повертає 0. Складність виконання алгоритму O(1).

Друга функція шукає даного футболіста в даному масиві футболістів з кількістю n. Якщо даний футболіст в масиві існує, то функція повертає індекс першого знайденого однакового футболіста. Якщо його в масиві немає (функція не може його знайти), повертає -1. Складність виконання алгоритму O(n).

```
Footbolist SetFootbolist() {
    Footbolist footbolist;
    cout << "Footbolist's surname: ";
    cin >> (&footbolist)->surname;
    cout << "Footbolist's role: ";
    cin >> (&footbolist)->role;
    cout << "Footbolist's age: ";
    cin >> (&footbolist)->age;
    cout << "Footbolist's games count: ";
    cin >> (&footbolist)->games_count;
    cout << "Footbolist's goal count: ";
    cin >> (&footbolist)->goal_count;
    return footbolist;
}
////////////////////////////////////////////////////////////////////////
```

Функція, яка зчитує інформацію з консолі про нового футболіста, і повертає його.

```
////////////////////////////////////////////////////////////////////////
Footbolist * ImportFootbolists(fstream* FFile, string filename, int * n){
    Footbolist* footbolist;
    (*FFile).open(filename);
    *FFile >> *n;
    footbolist = new Footbolist[*n];
    for (int i = 0; i < *n; i++) {
        *FFile >> (footbolist + i)->surname >> (footbolist + i)->role >>
            (footbolist + i)->age >> (footbolist + i)->games_count >>
            (footbolist + i)->goal_count;
    }
    (*FFile).close();
    return footbolist;
}
```

Функція, яка імпортує масив футболістів з їхньою кількістю з даного файлу. Функція задає значення n (кількість) і повертає вказівник на масив футболістів.

```
void ExportFootbolists(fstream* FFile, string filename, Footbolist* footbolist, int n) {
    (*FFile).open(filename, ofstream::out | ofstream::trunc);
    (*FFile) << n << " ";
    for (int i = 0; i < n; i++) {
        (*FFile) << (footbolist + i)->surname << " " << (footbolist + i)->role << " " <<
            (footbolist + i)->age << " " << (footbolist + i)->games_count << " " <<
            (footbolist + i)->goal_count;
    }
    (*FFile).close();
}
////////////////////////////////////////////////////////////////////////
```

Функція, яка експортує масив футболістів з їхньою кількістю в даний файл.

```
//////////////////////////////////////////////////////////////////
Footbolist* AddFootbolist(Footbolist* footbolist, int* n, Footbolist footbolist0) {
    Footbolist* newfootbolist = new Footbolist[((*n)+1)];
    for (int i = 0; i < *n; i++) {
        *(newfootbolist + i) = *(footbolist + i);
    }
    *(newfootbolist + *n) = footbolist0;
    (*n)++;
    return newfootbolist;
}
```

Функція, яка створює новий масив футболістів з кількістю n + 1, у який записує усіх футболістів попереднього масиву а також нового футболіста. Функція повертає вказівник на новий масив.

```
Footbolist* RemoveFootbolist(Footbolist* footbolist, int* n, Footbolist footbolist0) {
    if (*n > 0) {
        int k = FindFootbolist(footbolist, *n, footbolist0);
        if (k == -1) {
            cout << "Error! Requested footbolist can't be found" << endl;
            return footbolist;
        }
        else {
            int i;
            Footbolist* newfootbolist = new Footbolist[(*n) - 1];
            for (i = 0; i < k; i++) {
                *(newfootbolist + i) = *(footbolist + i);
            }
            for (i = k + 1; i < *n; i++) {
                *(newfootbolist + i - 1) = *(footbolist + i);
            }
            (*n)--;
            return newfootbolist;
        }
    }
    else return footbolist;
}
```

Функція, яка створює новий масив футболістів з кількістю n - 1, у який записує усіх футболістів попереднього масиву без заданого футболіста, але за умовою, якщо n > 0, та якщо заданий футболіст хоча б в одному екземплярі існує у масиві. Функція повертає вказівник на новий масив.

```
//////////////////////////////////////////////////////////////////
void FillFootbolists(Footbolist* footbolist, int n) {
    for (int i = 0; i < n; i++) {
        *(footbolist + i) = SetFootbolist();
    }
}
//////////////////////////////////////////////////////////////////
void PrintAll(Footbolist* footbolist, int n) {
    cout << n << endl;
    for (int i = 0; i < n; i++) PrintFootbolist(footbolist + i);
}
```

Перша функція заповнює масив футболістів з кількістю n інформацією.
Друга функція виводить масив футболістів з кількістю n у консоль.

```
void Menu(fstream* FFile, string filename) {
    cout << "0 for help" << endl;
    while (true) {
        int command = 0;
        cin >> command;
        if (command == 0) cout << "0 - help\n1 - new array\n2 - continue with existing\n3 - exit" << endl;
        else if (command == 3) {
            break;
        }
        else{
            int n = 0;
            if (command == 1) {
                cout << "Enter count: ";
                cin >> n;
                Footbolist* footbolist = new Footbolist[n];
                FillFootbolists(footbolist, n);
                ExportFootbolists(FFile, filename, footbolist, n);
                delete(footbolist);
            }
            else if (command == 2) {
                int i = 0;
                Footbolist* footbolist = ImportFootbolists(FFile, filename, &n);
                cout << "0 for help" << endl;
                while (true) {
                    cin >> command;
                    if (command == 0) cout << "0 - help\n1 - add footbolist\n2 - remove footbolist\n3 - find best forward\n4 - show all less than 5 games footbolist\n5 - print all\n6 - go back" << endl;
                    else if (command == 1) footbolist = AddFootbolist(footbolist, &n, SetFootbolist());
                    else if (command == 2) footbolist = RemoveFootbolist(footbolist, &n, SetFootbolist());
                    else if (command == 3) BestForward(footbolist, n);
                    else if (command == 4) LessThanFiveGames(footbolist, n);
                    else if (command == 5) PrintAll(footbolist, n);
                    else if (command == 6) {
                        break;
                    }
                    else cout << "Unknown command" << endl;
                    ExportFootbolists(FFile, filename, footbolist, n);
                }
                delete(footbolist);
            }
            else cout << "Unknown command" << endl;
        }
    }
}
```

Функція Menu приймає запити користувача та виконує їх.

```
int main()
{
    fstream FFile;
    Menu(&FFile, "practice1.txt");
}
```

Головна функція створює файловий потік і викликає функцію Menu, передаючи потік та назву файлу.