**Практична робота 2**

**Бурлаченко Єгор**

**Структури**

Код завдання 1:

#include <iostream>

#include <string>

using namespace std;

struct STUD {

string NAME;

string GROUP;

double SES[5];

};

void inputStudents(STUD students[], int N) {

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

cout << "Enter student " << i + 1 << " name: ";

cin >> students[i].NAME;

cout << "Enter student " << i + 1 << " group: ";

cin >> students[i].GROUP;

cout << "Enter student " << i + 1 << " grades (5 grades separated by spaces): ";

for (int j = 0; j < 5; j++) {

cin >> students[i].SES[j];

}

}

}

void printStudentsWithHighAverage(STUD students[], int N) {

bool found = false;

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

double sum = 0;

for (int j = 0; j < 5; j++) {

sum += students[i].SES[j];

}

double average = sum / 5;

if (average > 4.0) {

cout << "Student: " << students[i].NAME << ", Group: " << students[i].GROUP << ", Average: " << average << endl;

found = true;

}

}

if (!found) {

cout << "No students with an average greater than 4.0 found." << endl;

}

}

int main() {

int N;

cout << "Enter the number of students: ";

cin >> N;

STUD\* students = new STUD[N];

inputStudents(students, N);

printStudentsWithHighAverage(students, N);

delete[] students;

return 0;

}

Код завдання 2:

#include <iostream>

#include <string>

using namespace std;

struct Factory {

string name;

int age;

string specialty;

double salary;

bool isLocksmith() {

return specialty == "locksmith";

}

bool isTurner() {

return specialty == "turner";

}

};

int main() {

int N;

cout << "Enter the number of factories: ";

cin >> N;

Factory\* factories = new Factory[N];

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

cout << "Enter factory " << i + 1 << " name: ";

cin >> factories[i].name;

cout << "Enter factory " << i + 1 << " age: ";

cin >> factories[i].age;

cout << "Enter factory " << i + 1 << " specialty: ";

cin >> factories[i].specialty;

cout << "Enter factory " << i + 1 << " salary: ";

cin >> factories[i].salary;

}

int locksmithCount = 0;

int turnerCount = 0;

cout << "Factories with average age > 35:" << endl;

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

if (factories[i].isLocksmith()) {

locksmithCount++;

}

if (factories[i].isTurner()) {

turnerCount++;

}

if (factories[i].age > 35) {

cout << "Factory " << i + 1 << ": " << factories[i].name << " (Age: " << factories[i].age << ")" << endl;

}

}

cout << "Locksmiths: " << locksmithCount << endl;

cout << "Turners: " << turnerCount << endl;

delete[] factories;

return 0;

}

Контрольні запитання:

1. Структури в мовах програмування - це складні дані, які можуть містити різні типи даних, об'єднані під однією назвою. Вони дозволяють створювати власні складні типи даних для зберігання і організації інформації. У мові C, структури оголошуються ключовим словом `struct`, наприклад:

struct Person {

char name[50];

int age;

};

В мові C++, це можна зробити без ключового слова `struct`:

struct Person {

char name[50];

int age;

};

2. Основна відмінність між оголошеннями структур в мовах C і C++ полягає в тому, що в мові C++ вам не потрібно використовувати ключове слово `struct` для оголошення структури. У мові C оголошення структури виглядає так:

struct Person {

char name[50];

int age;

};

У мові C++ ви можете просто написати:

struct Person {

char name[50];

int age;

};

3. Для оголошення статичного масиву структур ви можете використовувати такий синтаксис:

Person people[10]; // Оголошення масиву структур Person розміром 10

Для динамічного масиву структур, вам потрібно виділити пам'ять за допомогою `new` (C++) або `malloc` (C) і використовувати покажчики.

4. Ось приклад оголошення масиву змінних типу структура:

struct Point {

int x;

int y;

};

Point points[5]; // Оголошення масиву структур Point розміром 5

5. Для звернення до полів структури ви використовуєте оператор "крапка". Наприклад:

Point p;

p.x = 10; // Звернення до поля x структури Point

p.y = 20; // Звернення до поля y структури Point

6. Для присвоєння і порівняння структур використовуються стандартні операції присвоєння (`=`) і порівняння (`==`, `!=`). Операції виконуються поелементно для кожного поля структури. Особливість полягає в тому, що ці операції створюються автоматично компілятором, але їх можна перевизначити за допомогою перевантаження операторів (для C++).

7. Об'єднання (union) - це структури, де всі поля займають один і той самий блок пам'яті. Об'єднання використовуються, коли потрібно зберегти дані різних типів в одному і тому ж місці пам'яті. Наприклад:

union Data {

int i;

float f;

char c;

};

8. Бітові множини (bit fields) - це структури, де ви можете зазначити кількість бітів, які будуть виділені для кожного поля. Вони дозволяють економити пам'ять, особливо коли потрібно зберігати булеві значення або числа з обмеженим діапазоном. Наприклад:

struct Flags {

unsigned int flag1 : 1;

// 1 біт

unsigned int flag2 : 2; // 2 біта

unsigned int flag3 : 4; // 4 біти

};

9. Бітові множини використовуються для зберігання бітових флагів, налаштувань апаратних пристроїв, а також для оптимізації використання пам'яті.

10. Для отримання двійкового представлення деякої змінної, ви можете використовувати різні методи виводу на екран. Наприклад, в C++:

int number = 42;

cout << "Binary representation: " << bitset<sizeof(number) \* 8>(number) << endl;

Де `bitset` використовується для виводу числа у двійковому форматі. `sizeof(number)` дає розмір змінної у байтах, і ми множимо його на 8, щоб отримати кількість бітів у змінній.