МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №10

**«Структуры, перечисления, объединения»**

Выполнил:

студент 1 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверила:

Хацкевич М. В.

Брест 2022

**Цель работы:** Изучить синтаксис и правила работы со структурами. Реализовать программу с применением структур, перечислений и объединений.

**Порядок выполнения работы**

*Текст задания*

Создать тип структуры согласно варинту, организовать поля этой структуры так, чтобы они содержали объединение, перечисление (можно добавить дополнительные поля) и битовое поле.

Создать массив структур, содержащий информацию согласно варианту индивидуального задания.

Реализовать работу с массивом структур через меню: ввод данных в массив, вывод собержимого массива на экран, сортировка по одному полю, удаления записи по заданному значению поля, выборка записей согласно индивидуального задания.

*Вариант 22*

Структура содержит следующие поля: страна, названия города, население города, доля насления школьного возраста (в процентах). Написать функцию для нахождения страны с самым большим количеством детей.

**Блок-схема**



#include <iostream>

#include <string>

#include <map>

#include <math.h>

using namespace std;

struct city {

string country;

string title;

int people;

double teens\_part; //õðàíèì ÷èñëîâóþ ÷àñòü, âûâîäÿ ÷èñëî äîáàâëÿåì %

enum additionalEnum { First, Second };

union additionalUnion {

int unsigned natural;

int intger;

double rational;

};

int additionalBitField : 2;

};

string getCountryWithMaxTeens(struct city\* cities, int arr\_length);

void menu(struct city\* cities, int arr\_length);

void displayCity(struct city& city);

void addCity(string& country, string& title, int people, double teens\_part, struct city\* cities, int& arr\_length);

void sortByField(string field, struct city\* cities, int arr\_length);

int findIdByField(string field, string value, struct city\* cities, int arr\_length, int pos = 0);

int findIdByField(string field, int value, struct city\* cities, int arr\_length, int pos = 0);

int findIdByField(string field, double value, struct city\* cities, int arr\_length, int pos = 0);

void deleteById(int key, struct city\* cities, int& arr\_length);

int main()

{

struct city cities[100];

int arr\_length = 0;

menu(cities, arr\_length);

return 0;

}

void menu(struct city\* cities, int arr\_length) {

string input;

cout << "Working with menu:\n0 - exit\t1 - get country with max teens\n2 - input new city\t3 - display cities\n4 - sorting\t5 - delete city\n";

while (input != "0") {

getline(cin, input);

if (input == "1") {

if (arr\_length > 0)

cout << "Country with max teens: " << getCountryWithMaxTeens(cities, arr\_length) << endl;

else

cout << "There's no city yet.\n";

}

if (input == "2") {

cout << "Country:\n";

string country;

getline(cin, country);

printf("City:\n");

string title;

getline(cin, title);

cout << "Amount of people:\n";

int people;

cin >> people;

cout << "Teens in %:\n";

double teens\_part;

cin >> teens\_part;

addCity(country, title, people, teens\_part, cities, arr\_length);

}

if (input == "3") {

for (int i = 0; i < arr\_length; i++) {

displayCity(cities[i]);

}

if (arr\_length <= 0) {

cout << "There's no city yet.\n";

}

}

if (input == "4") {

cout << "Type field for sorting (country, title, people, teens\_part):\n";

string field;

getline(cin, field);

while (!(field == "country" || field == "title" || field == "people" || field == "teens\_part")) {

cout << "Theres no such field!\n";

getline(cin, field);

}

sortByField(field, cities, arr\_length);

cout << "Done!\n";

}

if (input == "5") {

if (arr\_length <= 0) {

cout << "There's no city yet.\n";

}

else {

cout << "Type field for deleting (country, title, people, teens\_part):\n";

string field;

getline(cin, field);

while (!(field == "country" || field == "title" || field == "people" || field == "teens\_part")) {

cout << "Theres no such field!\n";

getline(cin, field);

}

cout << "Type value:\n";

int key;

if (field == "country" || field == "title") {

string value;

getline(cin, value);

key = findIdByField(field, value, cities, arr\_length, 0);

}

if (field == "people") {

int value;

cin >> value;

key = findIdByField(field, value, cities, arr\_length, 0);

}

if (field == "teens\_part") {

double value;

cin >> value;

key = findIdByField(field, value, cities, arr\_length, 0);

}

if (key >= 0) {

displayCity(cities[key]);

cout << "Got deleted!\n";

deleteById(key, cities, arr\_length);

}

else cout << "Value not found!\n";

}

}

}

}

string getCountryWithMaxTeens(struct city\* cities, int arr\_length) {

map<string, int> countries;

for (int i = 0; i < arr\_length; i++) {

countries[cities[i].country] += (int) round(cities[i].people \* cities[i].teens\_part);

}

string ans;

int teens = -1;

for (auto country = countries.begin(); country != countries.end(); country++) {

if (country->second > teens) {

ans = country->first;

teens = country->second;

}

}

return ans;

}

void displayCity(struct city& city) {

cout << "Country: " << city.country << ", City: " << city.title << ", People: " << city.people << ", Teens: " << city.teens\_part << "%\n";

}

void addCity(string& country, string& title, int people, double teens\_part, struct city\* cities, int& arr\_length) {

arr\_length++;

cities[arr\_length - 1].country = country;

cities[arr\_length - 1].title = title;

cities[arr\_length - 1].people = people;

cities[arr\_length - 1].teens\_part = teens\_part;

}

void sortByField(string field, struct city\* cities, int arr\_length) {

for (int i = 0; i < arr\_length; i++) {

for (int j = 0; j < arr\_length - 1; j++) {

if (field == "country")

if (cities[j].country > cities[j + 1].country) {

swap(cities[j], cities[j + 1]);

}

if (field == "title")

if (cities[j].title > cities[j + 1].title) {

swap(cities[j], cities[j + 1]);

}

if (field == "people")

if (cities[j].people > cities[j + 1].people) {

swap(cities[j], cities[j + 1]);

}

if (field == "teens\_part")

if (cities[j].teens\_part > cities[j + 1].teens\_part) {

swap(cities[j], cities[j + 1]);

}

}

}

}

int findIdByField(string field, string value, struct city\* cities, int arr\_length, int pos) {

for (; pos < arr\_length; pos++) {

if (field == "country")

if (cities[pos].country == value) return pos;

if (field == "title")

if (cities[pos].title == value) return pos;

}

return -1;

}

int findIdByField(string field, int value, struct city\* cities, int arr\_length, int pos) {

for (; pos < arr\_length; pos++) {

if (field == "people")

if (cities[pos].people == value) return pos;

}

return -1;

}

int findIdByField(string field, double value, struct city\* cities, int arr\_length, int pos) {

for (; pos < arr\_length; pos++) {

if (field == "teens\_part")

if (abs(cities[pos].teens\_part - value) < 0.001) return pos;

}

return -1;

}

void deleteById(int key, struct city\* cities, int& arr\_length) {

for (int i = key; i < arr\_length; i++) {

cities[i] = cities[i + 1];

}

arr\_length--;

}

**Вывод:** был изучен синтаксис и принципы работы со структурами. Согласно данному заданию была реализована программа с использованием массива структур, перечислений и объединений.