**Лабораторна робота №7**

**Розробка та реалізація програм з використанням директив**

**препроцесора в додатках**

**Ціль роботи**: оволодіння навичками складання програм з використанням директив препроцесора в додатках

**Завдання**

**Завдання 7.1.** Визначити дію фрагмента програми. Визначити вміст файлу file.txt

#include <iostream>

using namespace std;

# define mkstr(s) #s

int main()

{

cout << mkstr(I love JavaScript);

return 0;

}

*Розв’язання*

Цей фрагмент програми містить макрофункцію, а також оператор #, що заміняет текст на текст в лавпах. Тому в результаті виконання программи будет виведенно:

**I love JavaScript**

**Завдання 7.2.** Скласти та виконати програму, яка обробляє інформацію

з файлу. Файл треба створити на основі завдання 4.2. Виконати запити,

описані в завданні 4.2.

*Розв’язання*

1. Постановка задачі: Скласти та виконати програму, яка обробляє інформацію з файлу. Файл треба створити на основі завдання 4.2. Виконати запити, описані в завданні 4.2.
2. Алгоритм розв’язання задачі:

Алгоритм розв’язання задачі можна представити у вигляді такої послідовності дій:

1. Створити структуру countries
2. Зчитати усі країни з файлу.

- Формат даных:

- Кожний рядок – одна країна

- Порядок задання параметрів країни: Назва, Столиця, Населення, Площа

3. Виводимо список команд функцією printCommand

4. Запускаемо нескінченний цикл, що постійно запитуе у користувача номер команди

Можливі такі команди:

1. printAll (1) – вивід усіх країн в алфавітному порядку.
2. printByPopul (2) - вивід усіх країн по діапазону населення (min, max)
3. printBySAndPolul (3) – вивід усіх країн по діапазону площі з населенням більше вказанного числа.
4. Add (4) Додати країну
5. clearConsole(5) Очистити екран консоли (при цьому знову друкуется список команд)
6. exit (0) Вихід з програми.
7. Якщо введена неіснуюча программа, то має вивестись «Неправильная команда»
8. **Текст програми**

**main.cpp**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include <windows.h>

#include "List.h"

#include "functions.h"

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string adressDB = "input.txt";

auto \*countries = getCountriesFromFile(adressDB);

printCommand();

while (true) {

short n = prompt<short int>("> ");

if (n == 1) printAll(countries);

else if (n == 2) printByPopul(countries);

else if (n == 3) printBySAndPolul(countries);

else if (n == 4) add(countries, adressDB);

else if (n == 5) clearConsole();

else if (n == 0) {

exit();

break;

}

else cout << "Неправильная команда" << endl;

cout << endl;

}

return 0;

}

**functions.h**

#pragma once

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

#include <string>

using namespace std;

struct country {

string name;

string capital;

long long population;

int S;

};

/\*\*

\* Запрашивает от пользователя значение нужного типа

\* @param{char[]} label - текст, предложенный пользователю

\*

\* Привет работы:

\* prompt<int>("Введите целое число: ");

\* prompt<char>("Введите символ: ");

\* prompt<string>("Введите строку: ");

\*/

template <typename T>

T prompt(const char label[]) {

cout << label;

while (true) {

T val;

cin >> val;

if (cin.fail()) {

cin.clear();

cin.ignore(32767, '\n');

cout << "Произошла ошибка. Введите еще раз: ";

}

else {

cin.ignore(32767, '\n');

return val;

}

}

}

/\*\*

\* Вводит данные из файла в country

\* Возвращает true - если удалось испешно

\* @param{country \*&item} - country, в которую нужно записать данные

\*/

bool enterCountryFromFile(country \*&item, ifstream &fin) {

item = new country;

return bool(

fin >> item->name

>> item->capital

>> item->population

>> item->S

);

}

//Вставляет элемент списка в список

//После первого элемента, что меньше этого

//(елси для добавления использовать только

//эту функцию, то элементы будут отсортированы)

void insertSortToList(List<country\*> \*countries, country \*next) {

if (!countries->first()) {

countries->push(next);

}

else {

countries->forEach([&countries, &next](auto \*item) -> bool {

if (!item->next || item->next->val->name > next->name) {

countries->insertAfter(item, next);

return false;

}

return true;

}, countries->head);

}

}

/\*\*

\* Считывает странны из файла, и возвращает указатель на список указателей на странны

\* @param {string} adressDB - пусть к файлу со странами

\*/

auto getCountriesFromFile(string adressDB) {

ifstream fin(adressDB);

auto \*countries = getList<country\*>();

country \*next = NULL;

while (enterCountryFromFile(next, fin)) {

insertSortToList(countries, next);

}

fin.close();

return countries;

}

void printHeader() {

cout << "+-----------------+-----------------+-----------------+-----------------+" << endl

<< "| Страна | Столица | Население | Площадь |" << endl

<< "+=================+=================+=================+=================+" << endl;

}

void printCountry(country \*c) {

cout << left << "| " << setw(15) << c->name

<< " | " << setw(15) << c->capital

<< " | " << setw(15) << c->population

<< " | " << setw(15) << c->S << " |" << endl;

cout << "+-----------------+-----------------+-----------------+-----------------+" << endl;

}

//Выводит в консоль список команд

void printCommand() {

/\*#ifndef IS\_DEV

cout << "(Версия для разработки)" << endl << endl;

#endif\*/

cout << "Выберите команду: " << endl

<< "1. Вывести все страны" << endl

<< "2. Вывести страны с населением от min до max" << endl

<< "3. Вывести страны с площей от min до max и населением больше n" << endl

<< "4. Добавить страну" << endl

<< endl

<< "5. Очистить экран" << endl

<< "0. Выход" << endl << endl;

}

// - - - Menu - - -

void add(List<country\*>\* countries, string adressDB) {

country \*added = new country;

added->name = prompt<string>("Имя: ");

added->capital = prompt<string>("Столица: ");

added->population = prompt<long long>("Население: ");

added->S = prompt<int>("Площадь: ");

insertSortToList(countries, added);

ofstream fout(adressDB, ios\_base::app);

fout << endl <<

added->name << ' ' <<

added->capital << ' ' <<

added->population << ' ' <<

added->S;

fout.close();

}

void printAll(List<country\*>\* countries) {

cout << "Все страны: " << endl;

printHeader();

countries->forEach([](auto \*item) -> bool {

printCountry(item->val);

return true;

});

}

void printByPopul(List<country\*> \*countries) {

cout << "Страны с населением" << endl;

long long min = prompt<long long>("От: "),

max = prompt<long long>("До: ");

if (min > max) swap(min, max);

printHeader();

countries->forEach([min, max](auto \*item) -> bool {

long long popul = item->val->population;

if (popul >= min && popul <= max) {

printCountry(item->val);

}

return true;

});

}

void printBySAndPolul(List<country\*> \*countries) {

long long popul = prompt<long long>("Страны с населением больше: ");

int minS = prompt<int>("И площей от: ");

int maxS = prompt<int>("До: ");

if (minS > maxS) swap(minS, maxS);

printHeader();

countries->forEach([popul, minS, maxS](auto \*item) -> bool {

int S = item->val->S;

if (S >= minS && S <= maxS && item->val->population >= popul) {

printCountry(item->val);

}

return true;

});

}

void clearConsole() {

system("cls");

printCommand();

}

void exit() {

cout << "Завершение программы...";

}

#endif //FUNCTIONS\_H

**List.h**

#pragma once

#ifndef LIST\_H

#define LIST\_H

//Эллемент однонаправленного списка

template <typename T>

struct ListItem {

T val = NULL;

ListItem \*next = NULL;

};

//Однонаправленный список

template<typename T>

struct List {

ListItem<T> \*head, \*end;

//Добавляет элемента в конец списка

void push(T n) {

auto \*item = new ListItem<T>;

item->val = n;

end->next = item;

end = item;

}

//Добавляет элемент после указанного элемента

void insertAfter(ListItem<T> \*prev, T n) {

auto \*item = new ListItem<T>;

item->val = n;

item->next = prev->next;

prev->next = item;

};

template<typename \_Fn>

void forEach(\_Fn func, ListItem<T> \*cur = NULL) {

if (!cur) cur = first();

while (cur) {

bool answer = func(cur);

if (!answer) return;

cur = cur->next;

}

}

//Возвращает ссылку на первый реальный элемент

ListItem<T>\* first() {

return head->next;

}

//Иницилизирует список

List\* init() {

head = new ListItem<T>;

head->next = NULL;

end = head;

return this;

}

};

//Возвращает однонаправленный список

//Пример исопльзования:

//List<int> \*list = getList<int>();

template<typename T>

auto getList() {

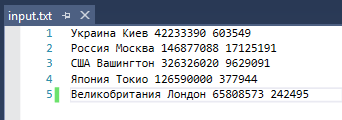
return (new List<T>)->init();

}

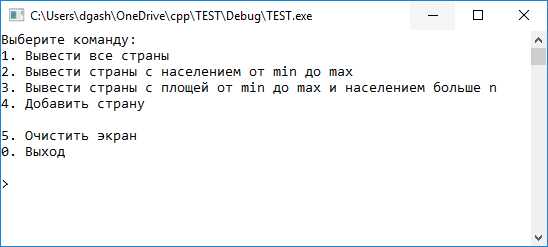
#endif //LIST\_H

Результат виконання програми:

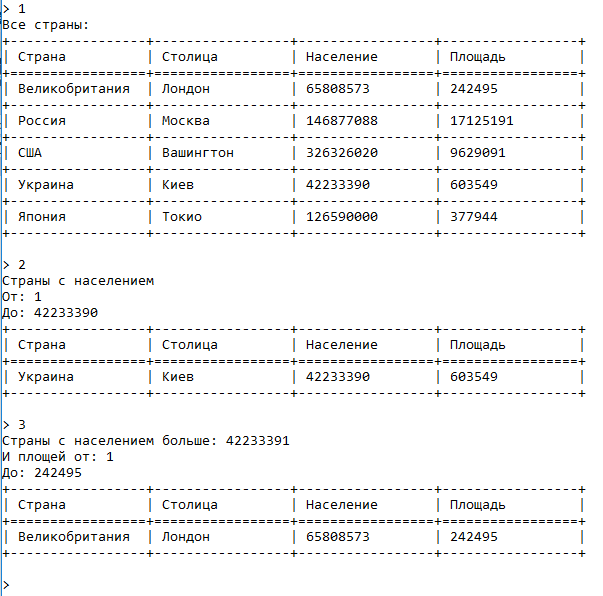
Input.txt в начале:



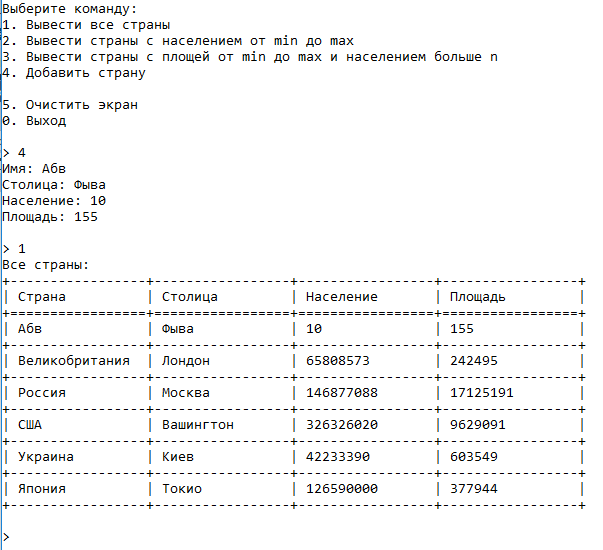
Приветствие:



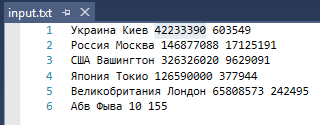
Демонстрация команд:



Добавление страны:



Input.txt после добавления страны



**Висновок:** на цій лабораторній роботі оволодів навичками складання програм з використанням директив препроцесора.