Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №13

за 2 семестр

По дисциплине: «ОАиП»

Тема: «Модульное программирование»

Выполнил:

Студентка 1 курса

Группы ПО-6(1)

Мартынович Д.

Проверил:

Хацкевич М. В.

2021

Лабораторная работа №13

Модульное программирование

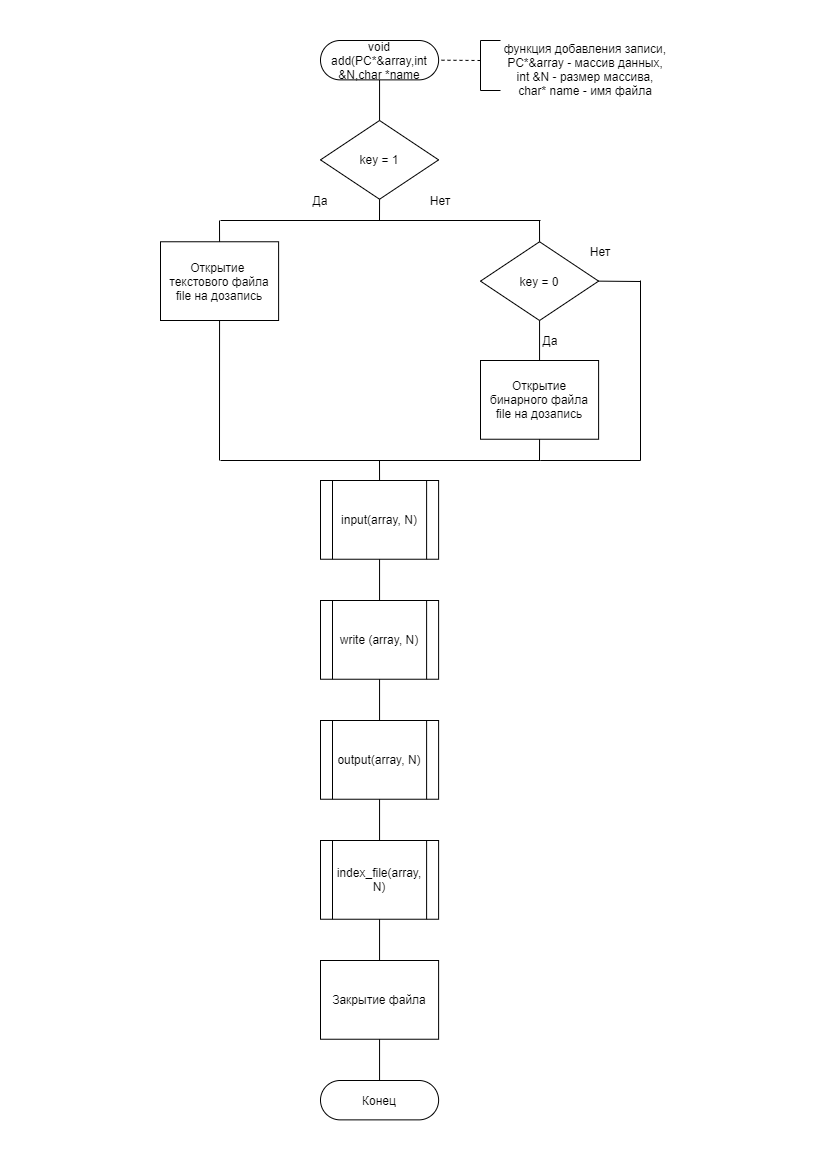
Цель работы: изучить принципы модульного программирования; ознакомиться с основными возможностями межмодульного взаимодействия.

**Задание:**

В программу, разработанную в лабораторной работе 12, внести следующие изменения и дополнения:

1. Программа должна быть разделена на несколько модулей (например, работа с файлами, работа с интерфейсом, обработка запросов к базе данных и т.п.).

2. Взаимодействие модулей организовать при помощи вызова функций и переменных внешнего типа.

Блок-схема: 

**Текст программы:**

1. **main.cpp**

#include <windows.h>

#include "menu.h"

int key;//переменная для определения с каким типом файлов необходимо работать

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("color F0");

char file\_name[20];

cout << "Работать с текстовым(1) или бинарным файлом(0)?:\n";

cin >> key;

cout << "Введите имя файла:\n";

cin.ignore();

gets\_s(file\_name);

if (key == 1) {

strcat\_s(file\_name, ".txt");

}

else if (key == 0) {

strcat\_s(file\_name, ".bin");

}

else {

cout << "Ошибка.\n";

}

system("cls");

menu(file\_name);

return 0;

}

1. **menu.h**

#pragma once

#include "interface.h"

void menu(char\* file\_name); //организация меню

**menu.cpp**

#include "menu.h"

void menu(char\* file\_name) {

int choise, N = 0;

PC\* array = new PC[N];

do {

cout << "Выберите действие:\n";

cout << "1) Ввод массива структур\n";

cout << "2) Вывод массива структур\n";

cout << "3) Сортировка массива структур \n";

cout << "4) Поиск процессора с наибольшим ядром\n";

cout << "5) Удаление заданной структуры\n";

cout << "6) Добавление новых элементов\n";

cout << "0) Выход\n";

cin >> choise;

system("cls");

switch (choise)

{

case 1:

read(array, N, file\_name);

output(array, N);

cin.ignore();

break;

case 2:

cin.ignore();

output(array, N);

break;

case 3:

cin.ignore();

sort(array, N, file\_name);

break;

case 4:

cin.ignore();

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

if (array[i].Yadra == search(array, N)) {

cout << "Процессор с самым большим ядром:\n";

cout << array[i].processor << " (" << array[i].Yadra << " )\n";

}

}

break;

case 5:

cin.ignore();

deleted(array, N, file\_name);

break;

case 6:

cin.ignore();

add(array, N, file\_name);

break;

default: break;

}

cin.ignore();

system("cls");

} while (choise != 0);

delete[] array;

}

1. **interface.h**

#pragma once

#include "text\_file.h"

void add(PC\*& array, int& N, char\* name); //добавление записей в конец

void sort(PC\* array, int N, char\* name); //сортировка

int search(PC\* array, int N); //поиск по заданному параметру

void deleted(PC\*& array, int& N, char\* name); //удаление записи

**interface.cpp**

#include "interface.h"

void sort(PC\* array, int N, char\* name) {

cout << "Выберите пункт:\n";

cout << "1 - Сортировка по названию \n";

cout << "2 - Сортировка по количеству ядер\n";

int choise;

cin >> choise;

for (int i = 0; i < N - 1; i++) {

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

if (choise == 1) {

if (strcmp(array[array[j].num].processor, array[array[i].num].processor) < 0) {

swap(array[i].num, array[j].num);

}

}

else if (choise == 0) {

if (array[array[j].num].Yadra < array[array[i].num].Yadra) {

swap(array[i].num, array[j].num);

}

}

}

}

index\_file(N, array);

rewrite(array, N, name);

output(array, N);

cin.ignore();

}

int search(PC\* array, int N) {

int max = array[0].Yadra, max\_ind = 0;

for (int i = 1; i < N; i++) {

if (array[i].Yadra > max)

max = array[i].Yadra;

}

return max;

}

void deleted(PC\*& array, int& N, char\* name) {

char Name[20];

cout << "Введите название процессора, который хотите удалить.\n";

output(array, N);

gets\_s(Name);

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

if (strcmp(array[i].processor, Name) == 0) {

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

if (array[j].num > array[i].num) {

array[j].num--;

}

}

N--;

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

array[j] = array[j + 1];

}

i--;

}

}

PC\* tempArr = new PC[N];

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

tempArr[i] = array[i];

}

array = new PC[N];

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

array[i] = tempArr[i];

}

delete[] tempArr;

system("cls");

index\_file(N, array);

rewrite(array, N, name);

output(array, N);

}

void add(PC\*& array, int& N, char\* name) {

ofstream file;

if (key == 1) {

file.open(name, ios\_base::app);

}

else if (key == 0) {

file.open(name, ios\_base::app | ios\_base::binary);

}

input(array, N);

file << endl;

write(array, N, file);

file.close();

output(array, N);

index\_file(N, array);

}

1. **text\_file.h**

#pragma once

#include <fstream>

#include "console.h"

void read(PC\*& array, int& N, char\* name); //чтение из файла

void write(PC\* array, int N, ofstream& file); //запись данных в файл

void rewrite(PC\* array, int N, char\* name); //перезапись данных в файле

void index\_file(int N, PC\* index); //индексные файлы

**text\_file.cpp**

#include "text\_file.h"

void rewrite(PC\* array, int N, char\* name) {

ofstream file;

if (key == 1) {

file.open(name);

}

else if (key == 0) {

file.open(name, ios\_base::binary);

}

write(array, N, file);

file.close();

}

void write(PC\* array, int N, ofstream& file) {

file << setw(15) << left << "Процессор" << setw(15) << left << "ГГц" << setw(15) << left << "Ядра" << setw(20) << left << "Память";

ifstream index\_file;

index\_file.open("Indexes.txt");

if (key == 1) {

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

int k;

if (index\_file.is\_open() && !index\_file.eof()) {

index\_file >> k;

index\_file >> k;

}

else k = i;

file << endl;

file << setw(15) << left << array[k].processor;

file << setw(15) << left << array[k].GHZ;

file << setw(15) << left << array[k].Yadra;

file << setw(15) << left << array[k].cache\_memory;

}

}

else if (key == 0) {

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

int k;

if (index\_file.is\_open()) {

index\_file >> k;

index\_file >> k;

}

else k = i;

file.write((char\*) & array[k], sizeof(array[k]));

}

}

index\_file.close();

}

void read(PC\*& array, int& N, char\* name) {

ifstream file;

if (key == 1) {

file.open(name);

}

else if (key == 0) {

file.open(name, ios\_base::binary);

}

N = 0;

if (!file.is\_open()) {

input(array, N);

rewrite(array, N, name);

}

else {

array = new PC[N];

char temp[1024];

file.getline(temp, 1024, '\n');

for (int i = 0; file.peek() != std::ifstream::traits\_type::eof(); i++) {

N++;

PC\* tempArr = new PC[N - 1];

for (int j = 0; j < N - 1; j++) {

tempArr[j] = array[j];

}

array = new PC[N];

for (int j = 0; j < N - 1; j++) {

array[j] = tempArr[j];

}

delete[] tempArr;

array[0].num = 0;

if (i >= 1)

array[i].num = array[i - 1].num + 1;

if (key == 1) {

file >> array[i].processor >> array[i].GHZ >> array[i].Yadra >> array[i].cache\_memory;

}

else if (key == 0) {

file.read((char\*) & array[i], sizeof(array[i]));

}

}

file.close();

}

}

void index\_file(int N, PC\* index) {

ofstream fout("Indexes.txt");

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

fout << i + 1 << " " << index[i].num;

if (i < N - 1)

fout << endl;

}

fout.close();

}

1. **console.h**

#pragma once

#include <iostream>

#include <iomanip>

using namespace std;

struct PC

{

char processor[20];

float GHZ;

int Yadra;

int cache\_memory;

int num;

union

{

int map;

int crash;

};

};

extern int key;

void input(PC\*& array, int& N); //ввод данных массива структуры

void output(PC\*& array, int N); //вывод данных массива структур

**console.cpp**

#include "console.h"

void input(PC\*& array, int& N) {

int n;

cout << "Введите количество структур: \n";

cin >> n;

N = N + n;

PC\* tempArr = new PC[N];

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

tempArr[i] = array[i];

}

array = new PC[N];

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

array[i] = tempArr[i];

}

delete[] tempArr;

system("cls");

cin.ignore();

array[0].num = 0;

for (int i = N - n; i < N; i++) {

if (i >= 1) {

array[i].num = array[i - 1].num + 1;

}

int choise;

cout << "1) Название процессора: ";

cin>>(array[i].processor);

cout << "2) Ггц: ";

cin>>(array[i].GHZ);

cout << "3) Ядра: ";

cin>>(array[i].Yadra);

cout << "4) Кэш-память: ";

cin>>(array[i].cache\_memory);

cin.ignore();

system("cls");

}

}

void output(PC \* &array, int N) {

cout << setw(15) << left << "№" << setw(15) << left << "Процессор" << setw(15) << left << "ГГц" << setw(15) << left << "Ядра" << setw(20) << left << "Память";

cout << endl;

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

cout << setw(15) << left << i + 1 << setw(15) << left << array[array[i].num].processor << setw(15) << left << array[array[i].num].GHZ << setw(15) << left << array[array[i].num].Yadra << setw(15) << left << array[array[i].num].cache\_memory;

cout << endl;

}

}

Вывод: изучил принципы модульного программирования и ознакомилcя с основными возможностями межмодульного взаимодействия.