

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет Информатики   
Кафедра Программных систем

**ОТЧЕТ**  
  
по лабораторному практикуму по дисциплине

**«Операционные системы»**

Вариант № 6

Студент Павленко В.Д.

Руководитель Востокин С.В.

Самара 2019

**Лабораторная работа №1/2**

**Цель лабораторных работ №№ 1, 2:** изучение синтаксиса языка Си. В лабораторной работе №1 реализуется алгоритм решения выбранной задачи. В лабораторной работе №2 решение оформляется в виде функций, выполняющих ввод данных пользователя с клавиатуры в консольном режиме, обработку данных согласно заданию и вывод результата.

**Вариант №4.** Напечатать все простые числа, не превосходящие заданное число M.

**Листинг (код программы):**

#include "pch.h"

#include <iostream>

#include <string.h>

#include <windows.h>

int simple(int num)

{

for (int i = 2; i <= num / 2; i++)

if ((num % i) == 0) return 0;

return 1;

}

int main()

{

setlocale(LC\_ALL, "Russian");

std::cout << "Программа выводит простые числа, не превосходящие число М\n";

std::cout << "М: ";

int m;

std::cin >> m;

std::cout << "\n";

for (int i = 2; i <= m; i++)

{

if (simple(i)) std::cout << i << " ";

}

std::cin >> m;

}

**Пример работы программы:**

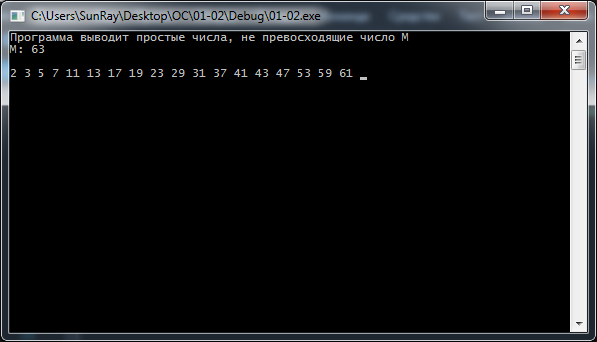


Рисунок 1

**Лабораторная работа №3**

**Цель лабораторной работы №3:** изучение методов работы с динамической памятью средствами программного интерфейса Win32. В первой части работы пишется программа с функциями создания динамического массива требуемого размера, обработки его согласно заданию, вывода и очистки с использованием функций библиотеки времени исполнения C/C++. Во второй части работы вызовы библиотеки времени исполнения заменяются вызовами программного интерфейса Win32.

**Вариант №4.** По динамическому массиву из M вещественных чисел необходимо рассчитать выборочное среднее и выборочную дисперсию.

**Листинг (код программы):**

#include "pch.h"

#include "windows.h"

#include <iostream>

int main()

{

setlocale(LC\_ALL, "Russian");

std::cout << "Программа находит среднее число и дисперсию ряда М чисел\n";

std::cout << "M: ";

int m;

std::cin >> m;

std::cout << "\n";

HANDLE hHeap;

hHeap = HeapCreate(HEAP\_GENERATE\_EXCEPTIONS, sizeof(float) \* m, 0);

if (hHeap != NULL)

{

float \*mas = (float\*)HeapAlloc(hHeap, 0, sizeof(float) \* m);

if (mas != NULL)

{

float sum = 0;

for (int i = 0; i < m; i++)

{

mas[i] = i + 1;

sum += mas[i];

std::cout << mas[i] << " ";

}

float average = sum / m;

sum = 0;

std::cout << "\n";

for (int i = 0; i < m; i++)

{

mas[i] -= average;

mas[i] \*= mas[i];

sum += mas[i];

std::cout << mas[i] << " ";

}

float disp = sum / m;

std::cout << "\nСреднее число: " << average << "\nДисперсия: " << disp << "\n";

}

else std::cout << "Память не выделена\n";

}

else std::cout << "Куча не создана\n";

HeapDestroy(hHeap);

system("pause");

}

**Пример работы программы:**

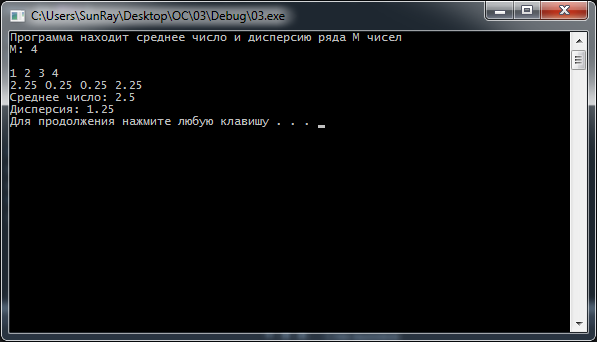


Рисунок 2

**Лабораторная работа №4/5/6**

**Цель лабораторной работы №4:** изучение методов работы с типами данных, определяемых пользователем на языке Си. Требуется реализовать в виде отдельной единицы компиляции (модуля) набор функций и объявлений данных, необходимых для работы с указанным в задании типом данных. В отдельном модуле пишется код для тестирования функций модуля.

**Цель лабораторной работы №5:** изучение функций ввода/вывода в программном интерфейсе Win32. Интерфейс модуля для работы с заданной структурой данных из задания №4 расширяется функциями для сохранения структуры данных на диске целиком и восстановления структуры данных из сохраненного файла.

**Цель лабораторной работы №6:** изучение методов работы с динамически подключаемыми библиотеками в программном интерфейсе Win32. Из модуля для работы с заданным типом данных, реализованным в задании №5, строится динамически подключаемая библиотека. Тестирующий код выполняет подключение библиотеки с использованием явного (парой вызовов LoadLibrary/GetProcAddress) и неявного (конфигурированием проекта) связывания.

**Вариант №4.**Требуется реализовать структуру данных «ассоциативный массив», используя хэш-таблицу, построенную по методу цепочек.

**Листинг (код программы):**

**04.cpp**

#include "pch.h"

#include <iostream>

#include <Windows.h>

#include <string>

using namespace std;

//Вариант 4: ассоциативный массив через хеш-таблицу методом цепочек

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

HINSTANCE hModule;

void(\*add)(string str, int value);

void(\*associative\_array)();

void(\*print)();

hModule = LoadLibrary(L"associative\_array.dll");

if (hModule != NULL)

{

associative\_array = (void(\*)())GetProcAddress(hModule, "??0associative\_array@@QAE@XZ"); //associative\_array

add = (void (\*) (string str, int value))GetProcAddress(hModule, "?add@associative\_array@@QAEXV?$basic\_string@DU?$char\_traits@D@std@@V?$allocator@D@2@@std@@H@Z"); //add

print = (void(\*) ())GetProcAddress(hModule, "?print@associative\_array@@QAEXXZ"); //print

if (associative\_array != NULL && add != NULL && print != NULL)

{

associative\_array();

add("строка1", 1);

add("строка1", 2);

add("строка2", 3);

print();

}

else cout << "\nFunctions weren't loaded";

}

else cout << "\nLibrary wasn't loaded";

FreeLibrary(hModule);

}

**Associative\_array.cpp**

#include "pch.h"

#include "associative\_array.h"

#include <string>

#include <iostream>

#include <functional>

#include <list>

#include <vector>

#include <Windows.h>

associative\_array::associative\_array()

{

arr = new std::vector<Pare>;

}

void associative\_array::add(std::string str, int value)

{

std::hash<std::string> hash;

int str\_hash = hash(str);

unsigned int i = 0;

Pare pare;

while (i < arr->size() && pare.key != str\_hash)

{

pare = arr->at(i);

i++;

}

if (pare.key == str\_hash)

{

arr->at(i - 1).list.push\_back(value);

}

else

{

arr->push\_back(Pare{ str\_hash, std::list<int>() });

arr->at(i).list.push\_back(value);

}

}

void associative\_array::del(std::string str, int value)

{

std::hash<std::string> hash;

int str\_hash = hash(str);

Pare pare;

bool end = false;

unsigned int i = 0;

while (i < arr->size() && !end)

{

if (str\_hash == arr->at(i).key)

{

arr->at(i).list.remove(value);

end = true;

}

i++;

}

}

void associative\_array::find(std::string str)

{

std::hash<std::string> hash;

int str\_hash = hash(str);

unsigned int i = 0;

bool end = false;

std::cout << "Результат поиска по запросу " << str << ": ";

while (i < arr->size() && !end)

{

if (str\_hash == arr->at(i).key)

{

auto iterator = arr->at(i).list.begin();

unsigned int j = 0;

while (iterator != arr->at(i).list.end())

{

std::cout << "\n " << \*iterator;

iterator++;

}

end = true;

std::cout << "\n";

}

i++;

}

if (!end)

std::cout << "не найдено\n\n";

}

void associative\_array::print()

{

unsigned int i = 0;

Pare pare;

while (i < arr->size())

{

pare = arr->at(i);

if (!(pare.list.empty()))

{

std::cout << "Ключ: " << arr->at(i).key << "\nСписок:\n";

auto iterator = pare.list.begin();

while (iterator != pare.list.end())

{

std::cout << " " << \*iterator << "\n";

iterator++;

}

std::cout << "\n";

}

i++;

}

}

void associative\_array::save()

{

//FILE\_SHARE\_WRITE | FILE\_SHARE\_READ в shareMode

HANDLE FileHandle = CreateFile(L"file.txt", GENERIC\_WRITE, 0, NULL, CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL, 0);

if (FileHandle == INVALID\_HANDLE\_VALUE)

std::cout << "INVALID\_HANDLE\_VALUE\n";

else

{

DWORD written;

unsigned int i = 0;

int tmp;

tmp = arr->size();

WriteFile(FileHandle, &tmp, sizeof(tmp), &written, NULL);

while (!arr->empty() && i < arr->size())

{

tmp = arr->at(i).key;

WriteFile(FileHandle, &tmp, sizeof(tmp), &written, NULL);

tmp = arr->at(i).list.size();

WriteFile(FileHandle, &tmp, sizeof(tmp), &written, NULL);

while (!arr->at(i).list.empty())

{

tmp = arr->at(i).list.front();

WriteFile(FileHandle, &tmp, sizeof(tmp), &written, NULL);

arr->at(i).list.pop\_front();

}

i++;

}

}

CloseHandle(FileHandle);

}

void associative\_array::load()

{

HANDLE FileHandle = CreateFile(L"file.txt", GENERIC\_READ, 0, NULL,

OPEN\_EXISTING, 0, NULL);

if (FileHandle != INVALID\_HANDLE\_VALUE)

{

arr->clear();

DWORD read;

BOOL RF;

unsigned int arrSize;

unsigned int listSize;

int key;

int tmp;

RF = ReadFile(FileHandle, &arrSize, sizeof(arrSize), &read, NULL);

if (RF && arrSize != 0)

{

for (int i = 0; i < arrSize; i++)

{

Pare pare;

arr->push\_back(pare);

ReadFile(FileHandle, &key, sizeof(key), &read, NULL);

arr->at(i).key = key;

ReadFile(FileHandle, &listSize, sizeof(listSize), &read, NULL);

for (int j = 0; j < listSize; j++)

{

ReadFile(FileHandle, &tmp, sizeof(tmp), &read, NULL);

arr->at(i).list.push\_back(tmp);

}

}

}

}

else std::cout << "File wasn't opened";

CloseHandle(FileHandle);

}

associative\_array::~associative\_array()

{

}

**Associative\_array.h**

#pragma once

#ifdef LISTDLL\_EXPORTS

#define DLL\_CALL \_\_declspec(dllexport)

#else

#define DLL\_CALL \_\_declspec(dllimport)

#endif

#include <string>

#include <list>

#include <vector>

class DLL\_CALL associative\_array

{

struct Pare {

int key;

std::list<int> list;

};

std::vector<Pare> \* arr;

public:

associative\_array();

void add(std::string, int);

void del(std::string, int);

void find(std::string);

void print();

void save();

void load();

~associative\_array();

};

**Пример работы программы:**

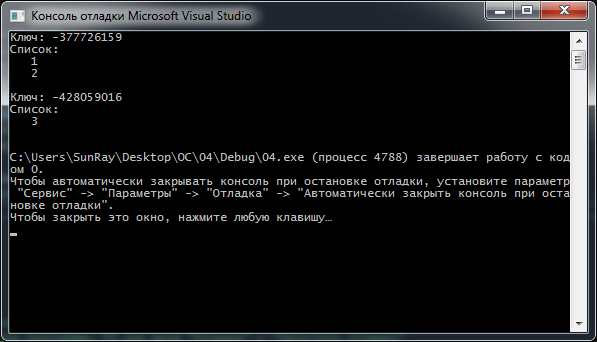


Рисунок 3