МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е.В. Павлов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «ОЦЕНКА КАЧЕСТВА ПРОГРАММНЫХ СРЕДСТВ» |
| по дисциплине: «МЕТРОЛОГИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ» |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4631 |  |  |  | С.А. Гришин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург   
2018

1. **Цель работы**

Целью данной работы является изучение оценочных показателей качества и получение навыков описания модели качества программного обеспечения.

1. **Задание**  
    Описать модель качества программного обеспечения при помощи оценочных показателей качества в соответствии со стандартом ГОСТ 28195-89.  
   В качестве варианта задания выбрана программа, написанная на языке программирования С++, [хеш-таблицу](https://github.com/gitGUAP/MPO/) с двойным хешированием.

|  |  |  |
| --- | --- | --- |
|  | Наименование показателя качества | Оценка, пояснение |
| 1. | Надежность |  |
| 1.1 | Возможность обработки ошибочных ситуаций | присутствует |
| 1.2 | Полнота обработки ошибочных ситуаций | присутствует |
| 1.3 | Наличие проверки допустимых значений входных данных | присутствует |
| 1.4 | Наличие системы контроля полноты входных данных | присутствует |
| 1.5 | Наличие средств контроля корректности входных данных | присутствует |
| 1.6 | Наличие средств восстановления процесса в случае сбоев оборудования | отсутствует |
| 1.7 | Наличие возможности разделения по времени выполнения отдельных функций программ | отсутствует |
| 1.8 | Наличие возможности повторного старта с точки останова | отсутствует |
| 1.9 | Наличие обработки неопределенностей (деление на 0, квадратный корень из отрицательного числа etc.) | отсутствует (не встречается) |
| 1.10 | Наличие централизованного управления процессами, конкурирующими из-за ресурсов | отсутствует (не встречается) |
| 1.11 | Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления | присутствует (обработка исключений) |
| 1.12 | Наличие средств, обеспечивающих завершение процесса в случае ошибок | присутствует |
| 1.13 | Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок | отсутствует |
| 2. | Сопровождаемость |  |
| 2.1 | Наличие комментариев в точках входа и выхода программы | отсутствует |
| 2.2 | Оценка простоты программы по числу точек входа и выхода: W = 1 / ((D +1) × (F + 1)), где D – общее число точек входа в программу, F – общее число точек выхода из программы | 1 / ((1 + 1) × (1 + 1)) = 0.25 |
| 2.3 | Оценка простоты программы по числу переходов по условию: U = (1 – А / В), где А – общее число переходов по условию,  В – общее число исполняемых операторов в программе | 1 – (13 / 25) = 0.48 |

1. **Оценка показателей качества программного обеспечения**

|  |  |  |
| --- | --- | --- |
| 2.4 | Оценка программы по числу циклов (количество циклов в программе) | 12 циклов for |
| 2.5 | Используется ли язык высокого уровня | да (C++) |
| 2.6 | Наличие заголовочных комментариев программы с указанием ее структурных и функциональных характеристик | отсутствует |
| 2.7 | Использование при построении программы метода структурного программирования | да |
| 2.8 | Использование при построении программы метода объектно- ориентированного программирования | да |
| 2.9 | Наличие ограничений на размеры модуля | отсутствует (ограничений нет) |
| 2.10 | Наличие модульной схемы программы и поддержка оверлейной структуры | отсутствует |
| 2.11 | Оценка программы по числу уникальных модулей | Все модули уникальны |
| 2.12 | Оценка программы по числу циклов с одним входом и одним выходом | Присутствуют двенадцать циклаов с одним входом и одним выходом |
| 3. | Удобство применения |  |
| 3.1 | Возможность освоения программных средств по документации | присутствует ( есть описание встроенных алгоритмов ) |
| 3.2 | Возможность освоения программы в обучающем режиме | отсутствует |

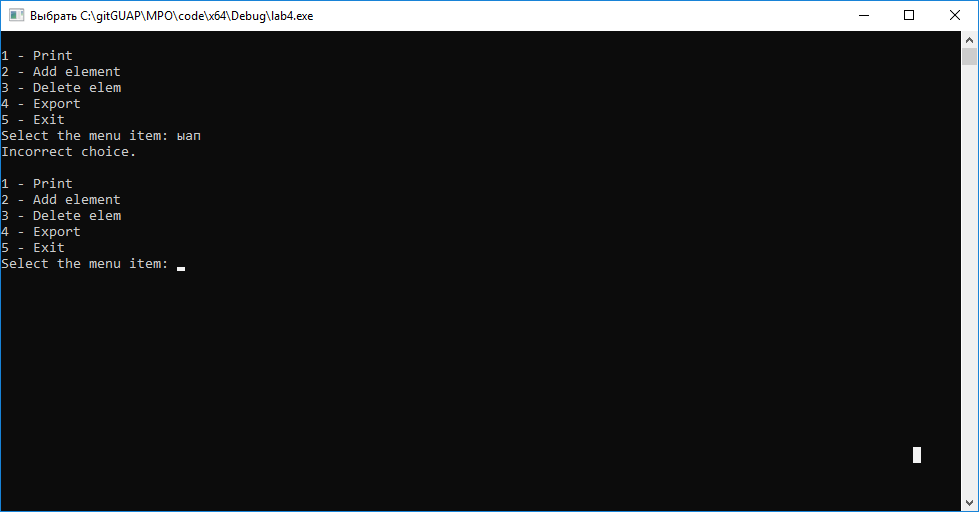
|  |  |  |
| --- | --- | --- |
| 3.3 | Полнота и понятность документации для освоения | отсутствует |
| 3.4 | Легкость и быстрота загрузки и запуска программы | присутствует |
| 3.5 | Легкость и быстрота завершения работы программы | отсутствует (прекращение работы программы возможно только при ручном завершении) |
| 3.6 | Возможность распечатки содержимого программы | отсутствует |
| 3.7 | Возможность приостанова и повторного запуска работы без потерь информации | отсутствует |
| 3.8 | Соответствие меню требованиям пользователя | меню присутствует |
| 3.9 | Возможность прямого перехода вверх и вниз по многоуровневому меню (пропуск уровней) | отсутствует |
| 3.10 | Возможность управления подробностью получаемых выходных данных | отсутствует |
| 3.11 | Обеспечение удобства ввода данных | присутствует |
| 3.12 | Интуитивно понятный интерфейс | присутствует |
| 3.13 | Легкость восприятия оперируемой информацией и данными | присутствует |
| 4. | Эффективность |  |
| 4.1 | Функции ввода/вывода | реализованы |
| 4.2 | Функции защиты и проверки данных | реализованы |
| 4.3 | Функции защиты от несанкционированного доступа | отсутствуют |
| 4.4 | Функции контроля доступа | отсутствуют |
| 4.5 | Число знаков после запятой в результатах вычислений | отсутствуют (не требуется) |
| 4.6 | Требуемый объем внутренней памяти (оперативная память) | >750 кб |
| 4.7 | Требуемый объем внешней памяти (дисковое пространство) | 190 кб |
| 4.8 | Оценка числа потенциальных пользователей | отсутствует |
| 4.9 | Оценка числа функций программного обеспечения | одна |
| 4.10 | Насколько набор функций удовлетворяет требованиям пользователя | полностью |
| 4.11 | Насколько возможности программ охватывают область решаемых пользователем задач | полностью |
| 4.12 | Возможность настройки формата выходных данных для конкретных пользователей | отсутствует |
| 4.13 | Оценка независимости модулей | подключаемые модули независимы, ПО может использоваться как модуль для другого проекта |
| 4.14 | Оценка программ по числу переходов и точек ветвления | в программе 13 точек ветвления |
| 4.15 | Оценка зависимости программы от программ операционной системы | не зависит |
| 4.16 | Зависимость от других программных средств | не зависит |
| 4.17 | Оценка программы по использованию условных переходов | 13 условных переходов |
| 4.18 | Оценка программы по использованию безусловных переходов | отсутствуют |
| 4.19 | Оценка программы по использованию локальных переменных | 1 локальная переменных |
| 4.20 | Оценка программы по числу комментариев | От 0 до 1 комментария к каждой функции |
| 4.21 | Комментарии к точкам ветвлений | отсутствуют |
| 4.22 | Комментарии к операторам объявления переменных | отсутствуют |
| 4.23 | Оценка семантики операторов | написаны в едином стиле |
| 4.24 | Семантика имен используемых переменных | написаны в едином стиле |
| 4.25 | Использование отступов, сдвигов и пропусков при формировании текста | присутствует |
| 5. | Корректность |  |
| 5.1 | Наличие интерфейса с пользователем | присутствует |
| 5.2 | Отсутствие противоречий в настройке системы | противоречия отсутствуют |
| 5.3 | Единообразие организации списков передаваемых параметров | все параметры передаются единообразно |
| 5.4 | Единообразие наименования каждой переменной и константы | все переменные и константы  названы единообразно;  наименования некоторых  переменных повторяются в  разных блоках кода (в разных функциях) |
| 5.5 | Используются ли разные идентификаторы для разных по смыслу переменных | используются |
| 5.6 | Все ли общие переменные объявлены как глобальные переменные | все |

1. **Выводы** В результате выполнения данной лабораторной работы были изучены оценочные показатели качества программного обеспечения в соответствии с ГОСТ 28195-89. Составлена модель качества на основе таких показателей, как надежность, сопровождаемость, удобство применения, эффективность и корректность.

Некоторые из показателей качества не были учтены должным образом или их оценка была произведена некорректно ввиду специфики оцениваемого программного обеспечения. В общем и целом, программа соответствует ГОСТ 28195-89.

1. *Какого рода ошибки могут возникнуть в процессе работы программы и каким образом их можно обработать? Приведите примеры; отдельно укажите какие ошибки обрабатываются в вашей программе.*

Ошибка ввода данных и переполнения таблицы, когда нужно увеличить размер внутреннего массива. В первом случае можно запросить ввод повторно, а во втором можно выделить память под новый массив и перенести данные из старого. В программе обрабатывается неверный ввод с помощью повторного ввода. В случае нехватки размера внутреннего массива происходит вывод ошибки.



1. *Чем характеризуются методы структурного и объектно-ориентированного программирования? Если мы программу, написанную на объектно-ориентированном языке, перепишем на структурный язык программирования, что именно изменится?*

Структурное программирование характеризуется построение программы из блоков: ветвление, последовательности, циклы. ООП – представление программы в виде совокупности объектов ( HashNode, HashMap). Способ её представления, уйдём от объектов, которые вызывают методы друг друга на список функций в которые будут передаваться данные.

1. *Что понимается под точкой входа и выхода из программы? Приведите примеры для любого языка программирования.*

Точка входа – место, откуда начнется выполнение программы, выхода – место в программе, где произойдёт прекращение работы программы.

main() – точка входа

return 0; - точка выхода из программы.

1. **Использованные источники**1. Черников Б.В. Оценка качества программного обеспечения: Практикум: учебное пособие / Б.В. Черников, Б.Е. Поклонов / Под ред. Б.В. Черникова. – М.: ИД "ФОРУМ": ИНФРА-М, 2012. – 400 с.: ил.

2. ГОСТ Р ИСО/МЭК 25-10-2015. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE).Модели качества систем и программных продуктов. – М.: Стандартинформ, 2015. – 29 с.

**Приложение**

MAIN.CPP

#include "HashMap.h"

#include <iostream>

#include <string>

#include <time.h>

using namespace std;

const int N = 2500;

const int K = 3;

void menu(HashMap<string> &HMap) {

cout << endl

<< "1 - Print\n"

<< "2 - Add element\n"

<< "3 - Delete elem\n"

<< "4 - Export\n"

<< "5 - Exit\n"

<< "Select the menu item: ";

int choice;

cin >> choice;

cin.clear();

cin.ignore(cin.rdbuf()->in\_avail());

switch (choice) {

case 1: {

HMap.print();

menu(HMap);

break;

}

case 2: {

string key;

string val;

cout << "Key: ";

getline(cin, key);

if (key.length() == 6 && isupper(key[0]) && isupper(key[5]) &&

isdigit(key[1]) && isdigit(key[2]) && isdigit(key[3]) &&

isdigit(key[4])) {

cout << "Value: ";

getline(cin, val);

HMap.put(key, val);

}

menu(HMap);

break;

}

case 3: {

string key;

cout << "Key: ";

getline(cin, key);

if (key.length() == 6 && isupper(key[0]) && isupper(key[5]) &&

isdigit(key[1]) && isdigit(key[2]) && isdigit(key[3]) &&

isdigit(key[4])) {

cout << "Elements with collisions: " << HMap.del(key).size() << endl;

}

menu(HMap);

break;

}

case 4: {

HMap.excel(string("excel.txt"));

cout << "OK" << endl;

menu(HMap);

break;

}

case 5: {

break;

}

default:

cout << "Incorrect choice." << endl;

menu(HMap);

}

}

int main() {

srand(time(0));

HashMap<string> HMap(N);

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

string strKey(6, '0');

string strVal(10 + rand() % 70, '\0');

strKey[0] = 'A' + rand() % 26;

strKey[1] = '0' + rand() % 9;

strKey[2] = '0' + rand() % 9;

strKey[3] = '0' + rand() % 9;

strKey[4] = '0' + rand() % 9;

strKey[5] = 'A' + rand() % 26;

for (int i = 0; i < strVal.length(); i++)

strVal[i] = (char)'A' + rand() % 26;

HMap.put(strKey, strVal);

}

menu(HMap);

return 0;

}

HASHNODE.H

#pragma once

#include <string>

template <typename V>

class HashNode {

public:

HashNode(const std::string& key, const V& value) : key(key), value(value) {}

std::string getKey() const { return key; }

V getValue() const { return value; }

void setValue(V val) { HashNode::value = val; }

private:

std::string key;

V value;

};

HASHMAP.H

#pragma once

#include "HashNode.h"

#include <fstream>

#include <iostream>

#include <vector>

using namespace std;

template <typename V> class HashMap {

public:

HashMap(int N) {

MAP\_SIZE = N;

map = new HashNode<V> \*[N];

exprt = new int[N];

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

map[i] = nullptr;

exprt[i] = 0;

}

// Generate prime numbers for h2

// 2, 3, 5, 7, 11, 13, 17, 19...N

primes\_h2.push\_back(2);

for (int i = 3; i < N; i++) {

bool prime = true;

for (int j = 0; j < primes\_h2.size() && primes\_h2[j] \* primes\_h2[j] <= i;

j++) {

if (i % primes\_h2[j] == 0) {

prime = false;

break;

}

}

if (prime)

primes\_h2.push\_back(i);

}

};

~HashMap() {

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

delete map[i];

delete map;

delete exprt;

};

unsigned int h1(string &key) {

unsigned int h1 = 0;

const int p = 37;

unsigned int p\_pow = 1;

// h(S) = S[0] + S[1] \* P + S[2] \* P ^ 2 + S[3] \* P ^ 3 + ... + S[N] \* P^N

for (size\_t i = 0; i < key.length(); i++) {

h1 += (key[i] - '0' + 1) \* p\_pow;

p\_pow \*= p;

}

return h1;

}

unsigned int h2(string &key) {

unsigned int h2 = 0;

for (size\_t i = 0; i < key.length(); i++)

h2 += key[i] \* key[i];

// Take a random item

return primes\_h2[h2 % primes\_h2.size()];

}

void put(string key, V value) {

unsigned int h1 = HashMap::h1(key);

unsigned int h2 = HashMap::h2(key);

for (size\_t i = 0; i < MAP\_SIZE; i++) {

exprt[h1 % N] += 1;

if (map[h1 % N] == nullptr) {

map[h1 % N] = new HashNode<V>(key, value);

return;

}

else {

if (map[h1 % N]->getKey() == key)

map[h1 % N]->setValue(value);

h1 = (h1 + h2) % N;

}

}

cout << "OVERFLOW" << endl;

}

HashNode<V> \*seach(string &key) {

unsigned int h1 = HashMap::h1(key);

unsigned int h2 = HashMap::h2(key);

for (size\_t i = 0; i < MAP\_SIZE; i++) {

if (map[h1 % N] != nullptr && map[h1 % N]->getKey() == key)

return map[h1 % N];

else

h1 = (h1 + h2) % N;

}

return nullptr;

}

vector<HashNode<V> \*> del(string &key) {

unsigned int h1 = HashMap::h1(key);

unsigned int h2 = HashMap::h2(key);

vector<HashNode<V> \*> collision;

for (size\_t i = 0; i < MAP\_SIZE; i++) {

if (map[h1 % N]->getKey() == key) {

delete map[h1 % N];

map[h1 % N] = nullptr;

return collision;

}

else {

collision.push\_back(map[h1 % N]);

h1 = (h1 + h2) % N;

}

}

}

void excel(string &name) {

ofstream fout(name);

for (size\_t i = 0; i < MAP\_SIZE; i++)

fout << exprt[i] << endl;

fout.close();

}

void print() {

cout << endl;

for (size\_t i = 0; i < MAP\_SIZE; i++)

if (map[i] != nullptr)

cout << i << ": " << map[i]->getKey() << " - " << map[i]->getValue()

<< endl;

}

private:

HashNode<V> \*\*map;

int \*exprt;

int MAP\_SIZE;

vector<int> primes\_h2;

};