|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ**  ***ИУК «Информатика и управление»***

**КАФЕДРА** ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Оценка качества программного продукта»**

**ДИСЦИПЛИНА: «Основы программной инженерии»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-22Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Карпуткин А.Н)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Амеличев Г.Э.)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель работы:** изучение основных методов и подходов оценки качества программного продукта.

**Задачи:**

* Изучить основы метрической теории Холстеда;
* Для написанных ранее программ произвести расчёт количественных характеристик программ;
* Сравнить полученные

**Листинг программы:**

#include <iostream>

#include <iomanip>

#include <ctime>

#define RANDOM rand()% (up\_line - low\_line + 1) + low\_line

// Прототипы используемых функций

void menu(int\* [], int, int, int);

int\*\* CreateMatrix(int, int, int);

void DeleteMatrix(int\* [], int);

void PrintMatrix(int\* [], int);

void MaxElements(int\* [], int, int, int);

void SumElements(int\* [], int);

void FirstDenied(int\* [], int);

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

bool rep = true;

while (rep == true) {

int len{};

cout << "Введите размер матрицы: ";

cin >> len;

int lowLine{}, upLine{};

cout << "Введите границы диапазона значений матрицы: ";

cin >> lowLine >> upLine;

int\*\* matrix = CreateMatrix(len, lowLine, upLine);

cout << endl;

menu(matrix, len, lowLine, upLine);

DeleteMatrix(matrix, len);

cout << endl << "Если вы хотите начать работу с новой матрицей, введите 1. Если хотите выйти, введите 0: ";

cin >> rep;

system("cls");

}

return 0;

}

void menu(int\* matrix[], int len, int lowLine, int upLine)

{

char var{};

bool exit = true;

while (exit == true) {

PrintMatrix(matrix, len);

cout << "Выберите желаемый пункт меню для работы с матрицей:" << endl << endl;

cout << " a) Найти максимальный элемент каждой строки и строку, которая содержит наименьший максимальный элемент." << endl;

cout << " b) Найти сумму модулей элементов в каждой строке, расположенных после первого отрицательного элемента в строке." << endl;

cout << " c) Найдите первый отрицательный элемент среди элементов матрицы, выделенных чёрным цветом (см. рис.)." << endl;

cout << " d) Выход из меню." << endl;

cout << endl << "Ваш выбор: ";

cin >> var;

switch (var) {

case 'a':

MaxElements(matrix, len, lowLine, upLine);

break;

case 'b':

SumElements(matrix, len);

break;

case 'c':

FirstDenied(matrix, len);

break;

case 'd':

break;

default:

cout << endl << "! Неверный ввод !" << endl;

}

cout << endl << "Если хотите продолжить работу, введите 1. Если хотите выйти из меню, введите 0: ";

cin >> exit;

system("cls");

}

}

// Создание динамической квадратной матрицы

int\*\* CreateMatrix(int len, int low\_line, int up\_line)

{

srand(time(0));

int\*\* matrix = new int\* [len] {};

for (int i{}; i < len; i++) {

matrix[i] = new int[len] {};

}

for (int i{}; i < len; i++) {

for (int j{}; j < len; j++) {

matrix[i][j] = RANDOM;;

}

}

return matrix;

}

// Вывод матрицы на экран

void PrintMatrix(int\* matrix[], int len)

{

cout << "Ваша матрица:" << endl;

for (int i{}; i < len; i++) {

for (int j{}; j < len; j++) {

cout << setw(5) << matrix[i][j];

}

cout << endl;

}

cout << endl;

}

// Поиск максимального элемента в каждой строке матрицы и позиции(строки) минимального элемента из этого набора

void MaxElements(int\* matrix[], int len, int low\_line, int up\_line)

{

int\* matr = new int[len] {};

cout << endl << "Максимальные элементы строк:" << endl;

for (int i{}; i < len; i++) {

int max = low\_line - 1;

for (int j{}; j < len; j++) {

if (matrix[i][j] > max) {

max = matrix[i][j];

}

}

matr[i] = max;

cout << matr[i] << " ";

}

int min = up\_line + 1;

int pos{};

for (int i{}; i < len; i++) {

if (matr[i] < min) {

min = matr[i];

pos = i;

}

}

cout << endl << endl << "Наименьший максимальный элемент находится в " << pos + 1 << " строке" << endl;

delete[] matr;

}

// В каждой строке ищется первое отрицательное число в строке и происходит сложение модулей всех последующих чисел в строке

void SumElements(int\* matrix[], int len)

{

int num = -1;

int sum{};

cout << endl;

for (int i{}; i < len; i++) {

for (int j{}; j < len; j++) {

if (matrix[i][j] < 0) {

num = j;

break;

}

}

if ((num == len - 1) || (num == -1)) {

sum = 0;

}

else {

for (int k = num + 1; k < len; k++) {

sum += abs(matrix[i][k]);

}

}

cout << "Сумма модулей для " << i + 1 << " строки:" << sum << endl;

num = -1;

sum = 0;

}

}

// Поиск первого отрицательного элемента в выделенной области матрицы

void FirstDenied(int\* matrix[], int len)

{

int denElement{};

for (int i{}; i < len && denElement >= 0; i++) {

for (int j = round(len / 3); j < len; j++) {

if (matrix[i][j] < 0) {

denElement = matrix[i][j];

break;

}

}

}

if (denElement < 0) {

cout << endl << "Первый отрицательный элемент в выделенной области матрицы: " << denElement << endl;

}

else {

cout << "Отрицательного элемента нет." << endl;

}

}

// Удаление динамической квадратной матрицы

void DeleteMatrix(int\* matrix[], int len)

{

for (int i{}; i < len; i++) {

delete[] matrix[i];

}

delete[] matrix;

cout << "Матрица успешно удалена." << endl;

}

**Измерение свойств алгоритма:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Операторы** | | | **Операнды** | | |
| **Номер** | **Оператор** | **Число вхождений** | **Номер** | **Операнд** | **Число вхождений** |
| **1** | () или {} | 114 | **1** | 1 | 10 |
| **2** | + | 7 | **2** | 0 | 8 |
| **3** | - | 6 | **3** | 3 | 1 |
| **4** | ; | 113 | **4** | i | 40 |
| **5** | = | 23 | **5** | j | 23 |
| **6** | > | 1 | **6** | k | 4 |
| **7** | [] | 40 | **7** | min | 3 |
| **8** | ++ | 14 | **8** | max | 4 |
| **9** | >= | 1 | **9** | up\_line | 4 |
| **10** | < | 18 | **10** | low\_line | 5 |
| **11** | >> | 6 | **11** | matrix | 26 |
| **12** | << | 54 | **12** | matr | 6 |
| **13** | cin | 5 | **13** | pos | 3 |
| **14** | main | 1 | **14** | num | 6 |
| **15** | cout | 12 | **15** | sum | 5 |
| **16** | for | 14 | **16** | len | 35 |
| **17** | rand | 1 | **17** | denElement | 5 |
| **18** | CreateMatrix | 1 | **18** | RANDOM | 2 |
| **19** | PrintMatrix | 1 | **19** | rep | 3 |
| **20** | MaxElements | 1 | **20** | var | 3 |
| **21** | SumElements | 1 | **21** | exit | 3 |
| **22** | FirsDenied | 1 |  |  |  |
| **23** | DeleteMatrix | 1 |  |  |  |
| **24** | new | 3 |  |  |  |
| **25** | delete | 3 |  |  |  |
| **26** | srand | 1 |  |  |  |
| **27** | time | 1 |  |  |  |
| **28** | using | 1 |  |  |  |
| **29** | return | 1 |  |  |  |
| **30** | endl | 26 |  |  |  |
| **31** | / | 1 |  |  |  |
| **32** | % | 1 |  |  |  |
| **33** | setw | 1 |  |  |  |
| **34** | break | 6 |  |  |  |
| **35** | abs | 1 |  |  |  |
| **36** | if | 6 |  |  |  |
| **37** | == | 4 |  |  |  |
| **38** | system | 2 |  |  |  |
| **39** | setlocale | 1 |  |  |  |
| **40** | menu | 3 |  |  |  |
| **41** | switch | 1 |  |  |  |
| **42** | case | 4 |  |  |  |
| **43** | default | 1 |  |  |  |
| **44** | else | 2 |  |  |  |
| **Итог 506** | | | **Итог 199** | | |

**Характеристики программы:**

Число различных операторов данной реализации: ;

Число различных операндов данной реализации: ;

Общее число всех операторов: ;

Общее число всех операндов: ;

Число различных входных и выходных операндов: ;

Словарь:

Длина реализации:

**Метрики Холстеда**

1. Длина программы: .
2. Объём программы: .
3. Потенциальный (минимальный) объем:
4. Граничный объем:
5. Соотношения между операциями и операндами (зависимость числа операндов от числа операций ): .
6. Уровень программы:
7. Сложность программы:
8. Оценка уровня программы:
9. Интеллектуальное содержание: \*
10. Работа по программированию (общее число элементарных мысленных различий, требуемых для порождения программы):
11. Приближенное время программирования:
12. Уровень языка:
13. Ожидаемое число ошибок:

**Вывод**: В ходе выполнения лабораторной работы были изучены основные методы и подходы оценки качества программного продукта, основы метрической теории Холстеда и произведён расчёт количественных характеристик программы на С++.