|  |  |
| --- | --- |
| **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**

**«Оценка качества программного продукта»**

**ДИСЦИПЛИНА: «Основы программной инженерии»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-22Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличев Г.Э. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

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

**Задачи:**

* изучить основы метрической теории Холстеда;
* Для написанных ранее программ произвести расчет количественных характеристик программ;
* Сравнить полученные результаты.

**Листинг:**

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

enum MenuCodes

{

PRINT\_ARRAY,

TASK\_1,

TASK\_2,

TASK\_3,

EXIT

};

const int MINIMUM\_RANDOM\_NUMBER = -12;

const int MAXIMUM\_RANDOM\_NUMBER = 12;

const int MINIMUM\_ARRAY\_BORDER = 3;

const int MAXIMUM\_ARRAY\_BORDER = 17;

void Menu();

void FillArray(int\*\* array, int arrayBorder, int mode);

void PrintArray(int\*\* array, int arrayBorder);

int SumOfDigits(int number);

int FindSideDiagonalSameRowNumber(int\*\* array, int lengthOfRow, int numberOfRow);

void IncreaseArrayElements(int\*\* array, int arrayBorder);

void CheckSignAlternation(int\*\* array, int arrayBorder);

void FindZeroElement(int\*\* array, int arrayBorder);

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

Menu();

return 0;

}

void Menu()

{

cout << "Введите размер квадратной матрицы (" << MINIMUM\_ARRAY\_BORDER << '-' << MAXIMUM\_ARRAY\_BORDER << "): ";

int arrayBorder{};

cin >> arrayBorder;

while (arrayBorder < MINIMUM\_ARRAY\_BORDER || arrayBorder > MAXIMUM\_ARRAY\_BORDER)

{

cout << "Неподходящее значение, попробуйте еще раз: ";

cin >> arrayBorder;

}

cout << "\nВыберете способ задания матрицы\n";

cout << "0. Вручную\n";

cout << "1. Случайно\n";

cout << "Выберете команду : ";

int mode{};

cin >> mode;

while (mode != 0 && mode != 1)

{

cout << "Неверная команда, попробуйте еще раз: ";

cin >> mode;

}

cout << '\n';

int\*\* array = new int\* [arrayBorder] {};

FillArray(array, arrayBorder, mode);

int code = -1;

while (code != EXIT)

{

cout << PRINT\_ARRAY << ". Вывести матрицу\n";

cout << TASK\_1 << ". Сложить элементы с суммой цифр элемента побочной диагонали той же строки\n";

cout << TASK\_2 << ". Проверка строки на чередование знаков\n";

cout << TASK\_3 << ". Поиск первого нулевого элемента\n";

cout << EXIT << ". Выход\n";

cout << "Выберете команду: ";

cin >> code;

while (code < PRINT\_ARRAY || code > EXIT)

{

cout << "Неверная команда, попробуйте еще раз: ";

cin >> code;

}

cout << "\n";

switch (code)

{

case PRINT\_ARRAY:

PrintArray(array, arrayBorder);

break;

case TASK\_1:

IncreaseArrayElements(array, arrayBorder);

break;

case TASK\_2:

CheckSignAlternation(array, arrayBorder);

break;

case TASK\_3:

FindZeroElement(array, arrayBorder);

break;

}

}

}

void FillArray(int\*\* array, int arrayBorder, int mode)

{

if (mode == 0)

{

cout << "Задайте матрицу\n";

}

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

{

array[i] = new int[arrayBorder] {};

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

{

if (mode == 0)

{

cin >> array[i][j];

}

else

{

array[i][j] = rand() % (MAXIMUM\_RANDOM\_NUMBER - MINIMUM\_RANDOM\_NUMBER + 1) + MINIMUM\_RANDOM\_NUMBER;

}

}

}

if (mode == 0)

{

cout << "\n";

}

}

void PrintArray(int\*\* array, int arrayBorder)

{

cout << " ";

for (int columnNumber = 0; columnNumber < arrayBorder; columnNumber++)

{

cout << setw(4) << columnNumber;

}

cout << "\n\n";

for (int row = 0; row < arrayBorder; row++)

{

cout << setw(2) << row << " ";

for (int column = 0; column < arrayBorder; column++)

{

cout << setw(4) << array[row][column];

}

cout << '\n';

}

cout << "\n";

}

int SumOfDigits(int number)

{

if (abs(number) < 10)

{

return abs(number);

}

else

{

return abs(number) % 10 + abs(number / 10);

}

}

int FindSideDiagonalSameRowNumber(int\*\* array, int lengthOfRow, int numberOfRow)

{

return array[numberOfRow][lengthOfRow - numberOfRow - 1];

}

void IncreaseArrayElements(int\*\* array, int arrayBorder)

{

for (int row = 0; row < arrayBorder; row++)

{

for (int column = 0; column < arrayBorder; column++)

{

if (column != arrayBorder - row - 1)

{

array[row][column] += SumOfDigits(FindSideDiagonalSameRowNumber(array, arrayBorder, row));

}

}

array[row][arrayBorder - row - 1] += SumOfDigits(array[row][arrayBorder - row - 1]);

}

}

void CheckSignAlternation(int\*\* array, int arrayBorder)

{

int row{};

cout << "Введите номер строки: ";

cin >> row;

if (row < 0 || row >= arrayBorder)

{

cout << "Ввод несуществующей строки\n";

return;

}

bool plusIsFound = false;

bool minusIsFound = false;

for (int column = 0; column < arrayBorder; column++)

{

if (array[row][column] > 0)

{

plusIsFound = true;

}

else if(array[row][column] < 0)

{

minusIsFound = true;

}

}

if (plusIsFound && minusIsFound)

{

cout << "Произошло чередование знаков\n";

}

else

{

cout << "Чередование знаков не произошло\n";

}

cout << "\n";

}

void FindZeroElement(int\*\* array, int arrayBorder)

{

int foundRow = -1;

int foundColumn = -1;

int firstRowOfSearch = arrayBorder / 3;

cout << "Первая строчка поиска: " << firstRowOfSearch << '\n';

for (int row = firstRowOfSearch; row < arrayBorder; row++)

{

for (int column = 0; column < arrayBorder; column++)

{

if (array[row][column] == 0)

{

foundRow = row;

foundColumn = column;

break;

}

}

if (foundRow != -1)

{

break;

}

}

if (foundRow != -1)

{

cout << "Первый нулевой элемент находится в " << foundRow << " строке " << foundColumn << " столбце\n";

}

else

{

cout << "Нулевых элементов в области поиска нет\n";

}

cout << "\n";

}

**Измерение свойств алгоритма:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Операторы** | | | **Операнды** | | |
| **Номер** | **Оператор** | **Число вхождений** | **Номер** | **Операнд** | **Число вхождений** |
| 1 | using | 1 | 1 | 0 | 19 |
| 2 | () или {} | 112 | 2 | 1 | 11 |
| 3 | [] | 23 | 3 | 2 | 1 |
| 4 | ; | 113 | 4 | 3 | 2 |
| 5 | , | 32 | 5 | 4 | 2 |
| 6 | : | 4 | 6 | 10 | 3 |
| 7 | “” | 34 | 7 | 12 | 2 |
| 8 | ‘’ | 4 | 8 | 17 | 1 |
| 9 | << | 53 | 9 | true | 2 |
| 10 | >> | 8 | 10 | false | 2 |
| 11 | < | 15 | 11 | i | 6 |
| 12 | > | 3 | 12 | j | 5 |
| 13 | >= | 1 | 13 | MINIMUM\_RANDOM\_NUMBER | 2 |
| 14 | == | 4 | 14 | MAXIMUM\_RANDOM\_NUMBER | 2 |
| 15 | != | 6 | 15 | MINIMUM\_ARRAY\_BORDER | 3 |
| 16 | = | 27 | 16 | MAXIMUM\_ARRAY\_BORDER | 3 |
| 17 | + | 3 | 17 | MenuCodes | 1 |
| 18 | ++ | 10 | 18 | PRINT\_ARRAY | 4 |
| 19 | += | 2 | 19 | TASK\_1 | 3 |
| 20 | - | 15 | 20 | TASK\_2 | 3 |
| 21 | / | 2 | 21 | TASK\_3 | 3 |
| 22 | % | 2 | 22 | EXIT | 4 |
| 23 | || | 3 | 23 | array | 30 |
| 24 | && | 2 | 24 | arrayBorder | 38 |
| 25 | abs | 4 | 25 | mode | 11 |
| 26 | if | 12 | 26 | number | 6 |
| 27 | else | 5 | 27 | lengthOfRow | 3 |
| 28 | for | 10 | 28 | numberOfRow | 4 |
| 29 | While | 4 | 29 | code | 7 |
| 30 | switch | 1 | 30 | columnNumber | 4 |
| 31 | case | 4 | 31 | row | 26 |
| 32 | break | 6 | 32 | column | 19 |
| 33 | return | 5 | 33 | plusIsFound | 3 |
| 34 | cout | 34 | 34 | minusIsFound | 3 |
| 35 | cin | 8 | 35 | foundRow | 5 |
| 36 | setw | 3 | 36 | foundColumn | 3 |
| 37 | new | 2 | 37 | firstRowOfSearch | 3 |
| 38 | rand | 1 | 38 | std | 1 |
| 39 | setlocale | 1 |  |  |  |
| 40 | srand | 1 |  |  |  |
| 41 | time | 1 |  |  |  |
| 42 | main | 1 |  |  |  |
| 43 | Menu | 3 |  |  |  |
| 44 | FillArray | 3 |  |  |  |
| 45 | PrintArray | 3 |  |  |  |
| 46 | SumOfDigits | 4 |  |  |  |
| 47 | FindSideDiagonalSameRowNumber | 3 |  |  |  |
| 48 | IncreaseArrayElements | 3 |  |  |  |
| 49 | CheckSighAlternation | 3 |  |  |  |
| 50 | FindZeroElement | 3 |  |  |  |
| **Итог 607** | | | **Итог 250** | | |

Число различных операторов

**Вывод:** в ходе работы были получены практические навыки работы в команде, использования стандарта кодирования, составления таблицы идентификаторов.