|  |  |
| --- | --- |
| **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 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_Янюк М. А.\_\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_Амеличев Г.Э.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2022 г. | | |

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 программы:**

1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
5. class Matrix {
6. public:
7. Matrix(const int \_size, const int \_startRange, const int \_endRange) {
8. m\_startRange = \_startRange;
9. m\_endRange = \_endRange;
10. if (\_size > M\_MMS) m\_size = M\_MMS;
11. else m\_size = \_size;
12. m\_matrix = new int\*[m\_size];
13. for (int i = 0; i < m\_size; i++) {
14. m\_matrix[i] = new int[m\_size];
15. }
16. }
18. ~Matrix() {
19. for (int i = 0; i < m\_size; i++) {
20. delete[] m\_matrix[i];
21. }
22. delete[] m\_matrix;
23. }
25. void fillMatrix() {
26. for (int i = 0; i < m\_size; i++) {
27. for (int j = 0; j < m\_size; j++) {
28. m\_matrix[i][j] = rand() % (m\_endRange - m\_startRange + 1) + m\_startRange;//!
29. }
30. }
31. }
33. void printMatrix() {
34. for (int i = 1; i < m\_size; i++) {
35. for (int j = 1; j < m\_size; j++) {
36. cout << setw(5) << m\_matrix[i][j];
37. }
38. cout << endl;
39. }
40. cout << endl;
41. }
43. void printRowSumInRange(const int \_leftBorder, const int \_rightBorder) {
44. m\_leftBorder = \_leftBorder;
45. m\_rightBorder = \_rightBorder;
46. for (int i = 0; i < m\_size; i++) {
47. int menuItem{};
48. for (int j = 0; j < m\_size; j++) {
49. if (m\_matrix[i][j] > m\_leftBorder and m\_matrix[i][j] < m\_rightBorder){
50. menuItem+=m\_matrix[i][j];
51. }
52. }
53. cout << endl << "In "<< i+1 << " row sum is "<< menuItem << endl;
54. }
55. }
57. void isPositiveRow() {
58. bool cur = 0;
59. for (int i = 0; i < m\_size; i++) {
60. bool lineFlag = 1;
61. if (!cur) {
62. for (int j = 0; j < m\_size; j++) {
63. if (lineFlag) {
64. if (m\_matrix[i][j] < 0) lineFlag = 0;
65. }
66. }
67. if (lineFlag) cur=1;
68. }
69. }
70. cout << endl << "There are "<< cur << " positive rows in the matrix"<< endl;
71. }
73. void getTriangleSum() {
74. int sum = 0;
75. for (int i = 0; i < m\_size; i++) {
76. for (int j = i; j < m\_size; j++) {
77. sum += m\_matrix[i][j];
78. }
79. }
80. cout << endl << "Sum in triangle is " << sum << endl;
81. }
83. private:
84. int m\_size;
85. // Range is for fillMatrix() method
86. int m\_startRange;
87. int m\_endRange;
88. // Borders is only for printRowSumInRange() method
89. int m\_leftBorder;
90. int m\_rightBorder;
91. int\*\* m\_matrix = nullptr;
92. const int M\_MMS = 100; // M\_MAXMATRIXSIZE
93. };
95. int InputMenu(){
96. cout << endl;
97. cout << "1- Для вывода матрицы" << endl;
98. cout << "2- Для проверки наличия положительных рядов" << endl;
99. cout << "3- Для нахождения суммы в диапазоне" << endl;
100. cout << "4- Для нахождения суммы выше главной диагонали" << endl;
101. cout << "0- Выход" << endl;
102. int menuNumber = 0;
103. cin >> menuNumber;
104. return menuNumber;
105. }
107. void MenuLoop(Matrix& Mat) {
108. int menuItem = -1;
109. int left = 0, right = 0;
110. while (menuItem != 0) {
111. menuItem = InputMenu();
112. switch(menuItem){
113. case 1:
114. Mat.printMatrix();
115. break;
116. case 2:
117. Mat.isPositiveRow();
118. break;
119. case 3:
120. cout << "Введите левую и правую границу: " ;
121. cin >> left >> right;
122. cout << endl;
123. Mat.printRowSumInRange(left, right);
124. break;
125. case 4:
126. Mat.getTriangleSum();
127. break;
128. case 0:
129. break;
130. default:
131. cout << "Неправильно выбрана команда, повторите ввод" << endl;
132. break;
133. }
134. }
135. }
137. int main() {
138. #pragma region init
139. srand(unsigned(time(NULL)));
140. Matrix Mat(18, -13, 13);
141. Mat.fillMatrix();
142. #pragma endregion init
144. MenuLoop(Mat);
146. return 0;
147. }

**Измерение свойств алгоритмов**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | операнды |  |  |  | операторы |  |
| Номер | операнд | Число вхождений |  | Номер | оператор | Число вхождений |
| 1 | M\_MMS | 56 |  | 1 | "=" | 36 |
| 2 | m\_startRange | 5 |  | 2 | ; | 89 |
| 3 | m\_endRange | 3 |  | 3 | () | 46 |
| 4 | m\_size | 2 |  | 4 | [] | 20 |
| 5 | Matrix | 34 |  | 5 | & | 1 |
| 6 | \_leftBorder | 11 |  | 6 | {} | 29 |
| 7 | \_rightBorder | 11 |  | 7 | << | 34 |
| 8 | cur | 2 |  | 8 | < | 14 |
| 9 | row | 32 |  | 9 | > | 2 |
| 10 | column | 25 |  | 10 | ++ | 12 |
| 11 | sum | 4 |  | 11 | + | 4 |
| 12 | 100 | 1 |  | 12 | % | 1 |
| 13 | 2 | 2 |  | 13 | >= | 1 |
| 14 | 1 | 13 |  | 14 | <= | 1 |
| 15 | 50 | 1 |  | 15 | -- | 1 |
| 16 | 0 | 22 |  | 16 | - | 4 |
| 17 | i | 24 |  | 17 | != | 1 |
| 18 | 18 | 1 |  | 18 | fillMatrix | 3 |
| 19 | 13 | 2 |  | 19 | printMatrix | 4 |
|  |  |  |  | 20 | printRowSumInRange | 3 |
| Итог |  | 248 |  | 21 | isPositiveRow | 3 |
|  |  |  |  | 22 | getTriangleSum | 3 |
|  |  |  |  | 23 | MenuLoop | 3 |
|  |  |  |  | 24 | InputMenu | 4 |
|  |  |  |  | 25 | main | 1 |
|  |  |  |  | 26 | srand | 1 |
|  |  |  |  | 27 | cout | 15 |
|  |  |  |  | 28 | endl | 16 |
|  |  |  |  | 29 | return | 3 |
|  |  |  |  | 30 | for | 14 |
|  |  |  |  | 31 | rand | 2 |
|  |  |  |  | 32 | setw | 1 |
|  |  |  |  | 33 | if | 6 |
|  |  |  |  | 34 | else | 1 |
|  |  |  |  | 35 | while | 1 |
|  |  |  |  |  |  |  |
|  |  |  |  |  | Итог | 380 |

**Расчетные характеристики:**

|  |  |  |
| --- | --- | --- |
| **Название** | **количество** | **Формула** |
| **Число уникальных операторов (n1)** | **35** |  |
| **Число уникальных операндов (n2):** | **19** |  |
| **Общее число операторов (N1):** | **380** |  |
| **Общее число операндов (N2):** | **248** |  |
| **словарь программы (n):** | **54** | **n = n1+n2** |
| **Экспериментальная длина программы (Nэ):** | **628** | **Nэ = N1+N2** |
| **Теоретическая длина программы (N' ):** | **260.2** | **N' =n1log2n1+n2log2n2** |
| **Объём программы (V):** | **3614.1** | **V = Nэlog2n** |
| **Потенциальный объём (V\*):** | **28.5** | **V\* = (2+n2\*)log2(2+n2\*)** |
| **Граничный объем (Vгр )** | **68.6** |  |
| **Уровень программы (L’):** | **0.008** | **L = V'/V** |
| **Сложность программы (S):** | **125** | **S = 1/L** |
| **Оценка уровня программы (L^):** | **0.004** | **L^ = (2n2)/(n1N2)** |
| **Интеллект программы (I):** | **29** | **I = L' \* V** |
| **Работа по программированию (Е):** | **458306** | **E = V^2/V'** |
| **Время программирования (T):** | **45830.6** | **T = E/10** |
| **Ожидание времени кодирования (T^):** | **25461.4** | **T' = E/S** |
| **Уровень языка программирования (л):** | **0.06** | **л = L^2\*V** |
| **Ожидаемое число ошибок (В)** | **1.2** | **B = V/E0** |

**Вывод:** в ходе выполнения лабораторной работы были изучены основные методы и подходы оценки качества программного продукта.