МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

**«НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ»**

**Кафедра програмного забезпечення комп'ютерних систем**

**Лабораторна робота №2**

з дисципліни: «ЯПЗ»

|  |  |
| --- | --- |
| Виконав: | ст. групи ПЗм-15-1м |
| Перевірив: | Козлов В.П. |

Дніпропетровськ

2016

**Тема:** Метрики сложности потока управления программ

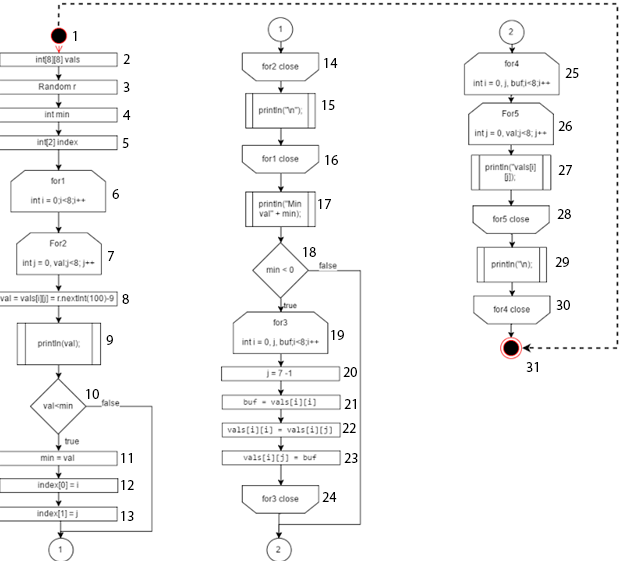
**Цель:** По полученному в лабораторной работе № 1 алгоритму рассчитать метрики сложности потока управления программ (метрики Маккейба, Джилба, и граничных значений).

**Задание:** 9. Ввести массив А(8, 8). Если минимальный элемент данного массива отрицателен, поменять местами главную и побочную диагонали массива. Вывести минимальный элемент, номера строки и столбца, в которых он находится, исходный и результирующий массивы.

**Исходный текст программы:**

//Массив исходных значений (матрица 8х8)  
**int**[][] vals = **new int**[8][8];  
Random r = **new** Random();  
**int** min = **Integer**.MAX\_VALUE;  
//Массив индексов минимального значения массива  
**int**[] index = **new int**[2];  
//Наполнение исходного/результирующего массивов  
//Поиск в массиве минимального значения  
**for** (**int** i = 0; i **<** 8; i**++**) {  
 **for** (**int** j = 0, val; j < 8; j++) {  
 val = vals[i][j] = r.nextInt( 100 ) - 9;  
 **System.out.print**(  
 ((val < 10 && val >= 0) ? ("0" + val) : val)  
 + (j == 7 ? "" : ", ") );  
 **if** ( val < min ) {  
 min = val;  
 //сохранение указателей на минимальное значение  
 index[0] = i;  
 index[1] = j;  
 }  
 }  
 **System.out.println**();  
}  
//Вывод минимального значения с указателями на него  
**System.out.println**( "\nMin value - \"" + min + "\" on position (x:y): " + index[1] + ":" + index[0] );  
  
**if** ( min < 0 ) {  
 **for** (**int** i = 0, j, buf; i < 8; i++) {  
 j = 7 - i; // указатель для побочной диагонали  
 buf = vals[i][i];  
 vals[i][i] = vals[i][j];  
 vals[i][j] = buf;  
 }  
}  
//Вывод результирующего массива  
**for** (**int** i = 0; i < 8; i++) {  
 **for** (**int** j = 0; j < 8; j++) {  
  **System.out.print**(  
 (vals[i][j] < 10 && (vals[i][j] >= 0) ? ("0" + vals[i][j]) : vals[i][j])  
 + (j == 7 ? "" : ", ") );  
 }  
 **System.out.println**();  
}

**Схема алгоритма:**



**Описание алгоритма:**

Задача программы выполнить поиск минимального значения матрицы и в случае если минимальное значение меньше нуля, то выполнить перемещения значений между основной и побочной диагоналями матрицы. Для этого выполнение программы можно разделить на 3 этапа:

1. Объявление и инициализация исходных данных, а именно переменные исходного массива, вспомогательные переменные для хранения минимального значения и его индексов.
2. Ввод значений матрицы и проверка введенного значения для поиска минимального. Вывод введенных данных на экран и найденного минимального значения.
3. Перемещение диагоналей матрицы в случае если найденное минимальное значение меньше нуля. Вывод результирующей матрицы.

**Расчет метрик потокоуправления:**

1. **Метрика Маккейба**

С помощью метрики Маккейба была проведена оценка трудоемкости тестирования программы. Метрика рассчитывается по формуле

где  **–** количество дуг графа, которое равняется **32,**

**–** количество вершин графа, которое равняется **31,**

**–** число компонентов связности графа, которое равняется **1.**

Подставив значение в формулу получаем:

Z(G) = 32 – 31 + 2\*1 = 3

Полученное значение показывает, что в работе алгоритма можно выделить 3 базовых линейно независимых контура:

1. 1 – 10(true) – 18(true) – 31;
2. 1 – 10(false) → 14 – 18(true) – 31;
3. 1 – 10(true) – 18(false) → 25 – 31;
4. **Метрика Джилба**

Расчет по метрике Джилба, определяет логическую сложность алгоритма(cl) по насыщенности условными операторами.

Результат метрики – соотношение количества условных операторов(СL= 2) к общем числу операторов программы (N = 31). CLI = 2.

Коэффициент логической сложности по метрике Джилба равен 0,1.

1. **Коэффициент скорректированной сложности**

Для нахождения коэффициента скорректированной сложности вершин графа было образовано 2 подграфа, начинающиеся с вершин выбора. В таблице 1.1 приведены расчеты по нахождению скорректированных сложностей вершин графа.

Таблица 1.1 – Скорректированные сложности вершин графа

|  |  |  |
| --- | --- | --- |
|  | Номер вершины выбора | |
| **Свойства подграфов** | **10** | **18** |
| **Номера вершин перехода** | 11,14 | 19,25 |
| **Номера вершин подграфа** | 11,12,13 | 19-24 |
| **Нижня граница подграфа** | 14 | 25 |
| **Сокректированая сложность вершины выбора** | 4 | 7 |

Абсолютной граничной сложностью графа является сумма скорректированных сложностей вершин этого графа. В таблице 1.2 приведены расчеты по нахождению коэффициента абсолютной граничной сложности программы.

Таблица 1.2 – Абсолютная граничная сложность графа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер вершины | 1-9 | 10 | 11-17 | 18 | 19-31 |  |
| Скорректированная сложность | 9 | 4 | 7 | 7 | 13 | 40 |

Подставив в формулу относительной граничной сложности графа полученные данные, получаем:

В таблице 2 приведены результаты расчетов метрик сложности потокоуправления программы для исходного графа.

Таблица 2 – Полученные коэффициенты метрик сложности потокоуправления

|  |  |
| --- | --- |
| Метрики сложности потокоуправления | |
| Метрика Маккейба - **Z(G)** | 3 |
| Абсолютная сложность программы - **CL** по **метрике Джилба** | 2 |
| Относительная сложность программы - **cl**  по **метрике Джилба** | 0,1 |
| Максимальный уровень вложенности условного оператора - **CLI** по **метрике Джилба** | 2 |
| Метрика граничных значений, **Sa** - абсолютная граничная сложность | 40 |
| Метрика граничных значений, **So** - относительная граничная сложность | 0,25 |

Выводы: в ходе выполнения лабораторной работы был проведен детальный анализ алгоритма и исходного кода программы по нескольким метрикам, которые указывают на сложность потокоуправления программы.