МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Кафедра «Информационных системы»

ОТЧЕТ

по лабораторной работе №1

Исследование способов анализа областей эквивалентности и структурного тестирования программного обеспечения, построения тестовых последовательностей

Выполнил:

ст. гр. ИС/б-31-о

Куркчи А. Э.

Проверил:

Строганов В. А.

Севастополь

2016

1. ЦЕЛЬ РАБОТЫ

Исследовать способы анализа областей эквивалентности входных данных для тестирования программного обеспечения, основные подходы к структурному тестированию программного обеспечения. Приобрести практические навыки составления тестовых последовательностей, построения графа потоков управления и определения независимых ветвей программы.

1. ПОСТАНОВКА ЗАДАЧИ
2. Написать программу, выполняющую заданные действия.
3. Определить области эквивалентности входных данных.
4. Построить граф потоков управления.
5. Вычислить цикломатическое число для построенного графа потоков управления.
6. Определить независимые ветви программы.
7. Составить примеры тестовых последовательностей.

Задача 1. Для целочисленной прямоугольной матрицы определить максимальное из чисел, встречающихся в заданной матрице один раз.

Задача 2. Дана строка. Подсчитать, сколько раз среди данных символов встречается буква а.

Задача 3. Программа, которая считывает из текстового файла пять предложений и выводит их в обратном порядке.

1. ХОД РАБОТЫ

Будем считать, что ввод данных для программ осуществляется идеальным пользователем через файл, имя которого передаётся первым параметром при запуске. При построении графа потоков проверку наличия первого параметра будем игнорировать.

1. Для целочисленной прямоугольной матрицы определить максимальное из чисел, встречающихся в заданной матрице один раз.
   1. Исходный код программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | #include <iostream> | | 2 | #include <fstream> | | 3 | #include <algorithm> | | 4 |  | | 5 | using namespace std; | | 6 |  | | 7 | *int* main(*int* *argc*, *char* \*\**argv*) { | | 8 | if(argc != 2) { | | 9 | cout << "Invalid usage" << endl; | | 10 | return -1; | | 11 | } | | 12 | ifstream is(argv[1]); | | 13 | *int* n, m; | | 14 | is >> n >> m; | | 15 | *int* len = n\*m; | | 16 |  | | 17 | *int* arr[len]; | | 18 | for(*int* i=0;i<len;i++) { | | 19 | is >> arr[i]; | | 20 | } | | 21 | sort(arr, arr+len); | | 22 |  | | 23 | *int* i = len -1; | | 24 | while(i > 0 && arr[i] == arr[i-1]) { | | 25 | *int* j = arr[i]; | | 26 | while(arr[i] == j) { | | 27 | i--; | | 28 | } | | 29 | } | | 30 |  | | 31 | if((i == 0 && (len == 1 || arr[i] != arr[i+1])) || i > 0) { | | 32 | cout << arr[i] << endl; | | 33 | } else { | | 34 | cout << "No such element" << endl; | | 35 | } | | 36 |  | | 37 |  | | 38 | return 0; | | 39 | } | |  |

* 1. Определение областей эквивалентности

По размерам матрицы:

1. В матрице только 1 элемент
2. В матрице несколько элементов
3. Матрица пуста

По значениям матрицы:

1. Все значения уникальны
2. Каждое значение встречается по меньшей мере два раза
3. Только одно значение не дублируется
   1. Построение графа потоков управления



Рисунок 1 – Граф потоков управления для задачи №1

* 1. Определение цикломатического числа

Число вершин: 10 + начало + конец = 13

Число дуг: 15

Цикломатическое число: 5

* 1. Определение независимых ветвей программы

1. 1 3 6 8 9
2. 1 2 1 3 4 3 6 7 9
3. 1 2 1 3 6 7 9
4. 1 2 1 3 4 5 4 3 6 7 9
5. 1 2 1 3 4 5 4 3 6 8 9
   1. Составление тестовых последовательностей

При составлении тестовых последовательностей по областям эквивалентности возможны следующие варианты: aa, ba, bb, bc, c

Независимая ветвь 2 невозможна, так как условие вложенного цикла таково, что в случае его проверки оно будет верным по крайней мере один раз.

Для ветви 1 соответствует тест c, для ветви 3 тест ba, для ветви 4 тест bc, а для ветви 5 тест b.

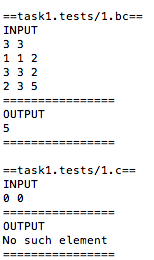
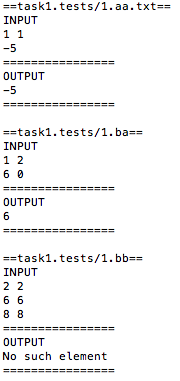


Рисунок 2 – Тестовые примеры для задачи №1

1. Дана строка. Подсчитать, сколько раз среди данных символов встречается буква а.
   1. Исходный код программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | #include <iostream> | | 2 | #include <fstream> | | 3 | #include <algorithm> | | 4 | #include <string.h> | | 5 |  | | 6 | using namespace std; | | 7 |  | | 8 | *int* main(*int* *argc*, *char* \*\**argv*) { | | 9 | if(argc != 2) { | | 10 | cout << "Invalid usage" << endl; | | 11 | return -1; | | 12 | } | | 13 | ifstream is(argv[1]); | | 14 |  | | 15 | *long* count = 0l; | | 16 | *char* c; | | 17 | while(is >> c) { | | 18 | if(c == 'a') { | | 19 | count++; | | 20 | } | | 21 | } | | 22 |  | | 23 | cout << count << endl; | | 24 |  | | 25 |  | | 26 | return 0; | | 27 | } | |  |

* 1. Определение областей эквивалентности

По размеру строки:

1. Пустая строка
2. Не пустая строка

По наличию целевой буквы:

1. Нет целевой буквы
2. Только одна целевая буква
3. Несколько целевых букв
   1. Построение графа потоков управления



Рисунок 3 – Граф потоков управления для задачи №2

* 1. Определение цикломатического числа

Число вершин: 4 + начало + конец = 6

Число дуг: 7

Цикломатическое число: 3

* 1. Определение независимых ветвей программы

1. 1
2. 1 2 1
3. 1 2 3 1
   1. Составление тестовых последовательностей

При составлении тестовых последовательностей по областям эквивалентности возможны следующие варианты: a, ba, bb, bc

Для ветви 1 соответствует тест a, для ветви 2 тест ba, а для ветви 3 тест bb и bc.

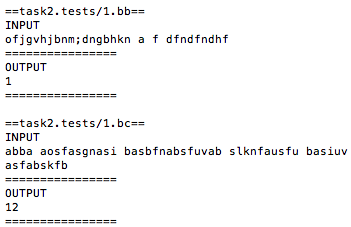
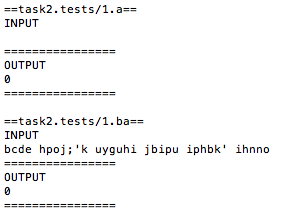


Рисунок 4 – Тестовые примеры для задачи №2

1. Программа, которая считывает из текстового файла пять предложений и выводит их в обратном порядке.
   1. Исходный код программы

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 | #include <iostream> | | 2 | #include <fstream> | | 3 | #include <algorithm> | | 4 | #include <string.h> | | 5 | #include <stack> | | 6 |  | | 7 | using namespace std; | | 8 |  | | 9 | *typedef* pair<*int*,*int*> pii; | | 10 |  | | 11 | *int* main(*int* *argc*, *char* \*\**argv*) { | | 12 | if(argc != 2) { | | 13 | cout << "Invalid usage" << endl; | | 14 | return -1; | | 15 | } | | 16 | ifstream is(argv[1]); | | 17 |  | | 18 | stack<pii> parts; | | 19 | *char* c; | | 20 | *int* i = 0; | | 21 | *int* start = 0; | | 22 | *bool* startSearch = false; | | 23 | is >> noskipws; | | 24 | while(is >> c) { | | 25 | if(startSearch) { | | 26 | if(c != ' ' && c != '\t' && c != '\n') { | | 27 | start = i; | | 28 | startSearch = false; | | 29 | } | | 30 | } else { | | 31 | if(c == '.' || c == '!' || c == '?') { | | 32 | parts.push(make\_pair(start, i)); | | 33 | startSearch = true; | | 34 | } | | 35 | } | | 36 | i++; | | 37 | } | | 38 | if(!startSearch) { | | 39 | parts.push(make\_pair(start,i-1)); | | 40 | } | | 41 |  | | 42 | pii el; | | 43 | while(parts.size()) { | | 44 | el = parts.top(); | | 45 | parts.pop(); | | 46 | is.clear(); | | 47 | is.seekg(el.first); | | 48 | for(*int* i=el.first;i <= el.second; i++) { | | 49 | is >> c; | | 50 | cout << c; | | 51 | } | | 52 | cout << " "; | | 53 | } | | 54 | cout << endl; | | 55 |  | | 56 | return 0; | | 57 | } | |  | |  |

* 1. Определение областей эквивалентности

По количеству предложений:

1. Ни одного предложения
2. Пять предложений
3. Несколько предложений

По наличию символов:

1. Нет пробельных символов
2. Есть пробельные символы
3. Есть восклицательные и вопросительные знаки
   1. Построение графа потоков управления



Рисунок 5 – Граф потоков управления для задачи №3

* 1. Определение цикломатического числа

Число вершин: 15 + начало + конец = 17

Число дуг: 23

Цикломатическое число: 9

* 1. Определение независимых ветвей программы

1. 1 7 9
2. 1 2 3 1 7 9 10 11 10 9
3. 1 2 3 1 7 8 9 10 11 10 9
4. 1 2 3 4 1 7 9 10 11 10 9
5. 1 2 3 4 1 7 8 9 10 11 10 9
6. 1 2 5 1 7 9 10 11 10 9
7. 1 2 5 1 7 8 9 10 11 10 9
8. 1 2 5 6 7 9 10 11 10 9
9. 1 2 5 6 7 8 9 10 11 10 9
   1. Составление тестовых последовательностей

При составлении тестовых последовательностей по областям эквивалентности возможны следующие варианты: a, ba, bb, bc, ca, cb, cc.

Активация вершин 3 и 5 в цикле всегда происходит поочерёдно, после активации вершины 5, в следующей итерации цикла будет затронута вершина 3 и ни в каком другом случае, потому все тесты, затрагивающие 5ю вершину справедливы и для проверки в 3ей вершине, что справедливо и наоборот Для каждой из независимых ветвей соответствует по меньшей мере один тест, получений с использованием областей эквивалентности.

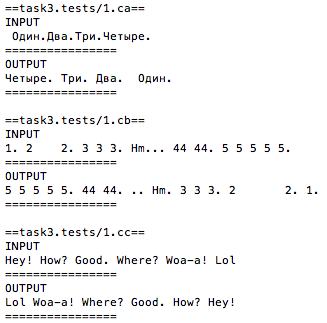
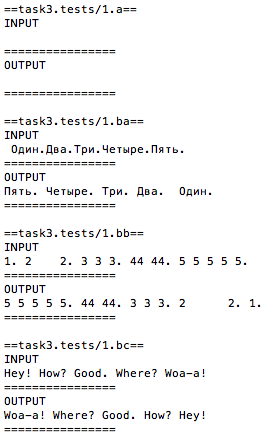


Рисунок 6 – Тестовые примеры для задачи №3

ВЫВОДЫ

В ходи данной лабораторной работы были исследованы способы анализа областей эквивалентности входных данных для тестирования программного обеспечения, основные подходы к структурному тестированию программного обеспечения. Были приобретены практические навыки составления тестовых последовательностей, построения графа потоков управления и определения независимых ветвей программы.