**Отчет по лабораторной работе № 11** по курсу “Фундаментальная информатика”

Студент группы М80-103Б-21 Березнев Никита Вадимович, № по списку 4

Контакты e-mail: nikita.berezneff@yandex.ru

Работа выполнена: «19» ноября 2021г.

Преподаватель: каф. 805 Севастьянов Виктор Сергеевич

Отчет сдан « » \_\_\_\_\_\_\_\_\_20\_\_ г., итоговая оценка \_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** Обработка последовательности литер входного текстового файла. Простейшие приемы лексического анализа. Диаграммы состояний и переходов.

1. **Цель работы:** Составить программу, выполняющую анализ и обработку вводимого ascii-текста. Требуется построчно напечатать значения пятеричных чисел, не являющихся троичными числами, в десятичной системе. При решении задачи необходимо использовать матаппарат конечных автоматов. Запрещено использовать массивы, указатели и явные числовые коды ascii-знаков (например, число 48 вместо литерной константы '0').
2. **Задание (вариант 9):** Составить программу, выполняющую анализ и обработку вводимого ascii-текста. Требуется построчно напечатать значения пятеричных чисел, не являющихся троичными числами, в десятичной системе.
3. **Оборудование** (студента):

Процессор *Intel Core i3-3217U @ 4x 1.80GH* с ОП *7851* Мб, НМД *512* Гб. Монитор *1366x768*

1. **Программное обеспечение (**студента**):**

Операционная система семейства: *linux*, наименование: *ubuntu*, версия *18.10 cosmic*

интерпретатор команд: *bash* версия *4.4.19*.

Система программирования -- версия --

Утилиты операционной системы:

Прикладные системы и программы --

Местонахождение и имена файлов программ и данных на домашнем компьютере --

**6. Идея, метод, алгоритм** решения задачи(в формах:словесной,псевдокода,графической[блок-схема,диаграмма,рисунок,таблица] или формальные спецификации с пред- и постусловиями).

- Использование конечных автоматов позволяет реализовать посимвольное считывание элементов файла для перехода между различными состояниями конечного автомата. Алгоритм переходов между состояниями:

- Начальное состояние автомата равно А. Если при считывании символ равен ‘0’, ‘1’, ‘2’, + то автомат переходит в состояние P0 (Положительное начальное). Если при считывании символ равен ‘3’, ‘4’, то автомат переходит в состояние P1 (Положительное конечное). Если при считывании символ равен ‘-‘, то автомат переходит в состояние N0 (Отрицательное начальное). Если при считывании символ равен ‘ ‘, ‘\n’, то происходит переприсваивание значений переменным, при этом, если считанный символ равен ‘\n’, то происходит переход на новую строку. Если считанный символ не равен ни одному из вышеперечисленных, то слово является некорректным и автомат переходит в состояние Bad. Если после этого состояние автомата != Bad, а также считанный символ != ‘-‘, ‘+’, ‘ ‘, ‘\n’, то считанный символ сохраняется, как одна из цифр числа.

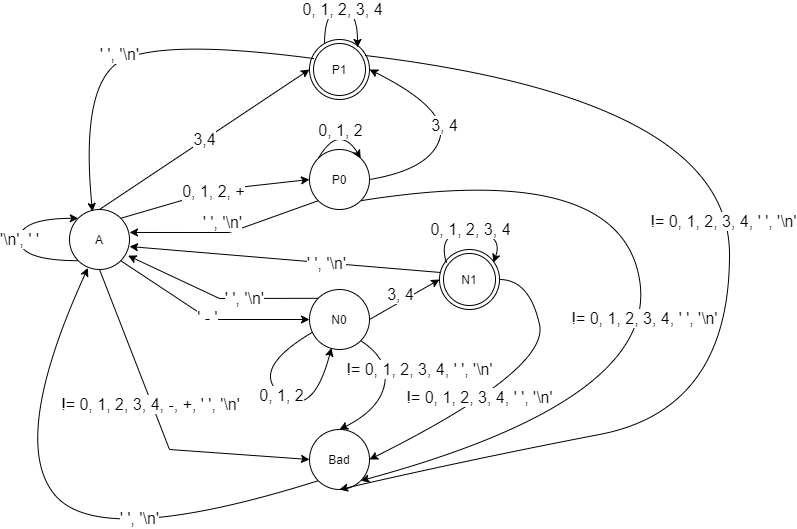
- Состояние P0. Если при считывании символ равен ‘3’, ‘4’, то автомат переходит в состояние P1 (Положительное конечное). Если при считывании символ равен ‘ ‘, ‘\n’, то происходит переприсваивание значений переменным, при этом, если считанный символ равен ‘\n’, то происходит переход на новую строку. Если считанный символ != ‘0‘, ‘1’, ‘2‘, ‘3’, ‘4’, ‘ ‘, ‘\n’, то слово является некорректным и автомат переходит в состояние Bad. Если после этого состояние автомата != Bad, а также считанный символ != ‘ ‘, ‘\n’, то считанный символ сохраняется, как одна из цифр числа.

- Состояние P1. Если при считывании символ равен ‘ ‘, ‘\n’, то происходит перевод числа, составленного из считанных цифр, в десятичный вид и осуществляется его вывод; автомат переходит в состояние А; происходит переприсваивание значений переменным. Если считанный символ != ‘0‘, ‘1’, ‘2‘, ‘3’, ‘4’, ‘ ‘, ‘\n’, то слово является некорректным и автомат переходит в состояние Bad. Если после этого состояние автомата != Bad, а также считанный символ != ‘ ‘, ‘\n’, то считанный символ сохраняется, как одна из цифр числа.

- Состояние N0. Если при считывании символ равен ‘3’, ‘4’, то автомат переходит в состояние N1 (Отрицательное конечное). Если при считывании символ равен ‘ ‘, ‘\n’, то происходит переприсваивание значений переменным, при этом, если считанный символ равен ‘\n’, то происходит переход на новую строку. Если считанный символ != ‘0‘, ‘1’, ‘2‘, ‘3’, ‘4’, ‘ ‘, ‘\n’, то слово является некорректным и автомат переходит в состояние Bad. Если после этого состояние автомата != Bad, а также считанный символ != ‘ ‘, ‘\n’, то считанный символ сохраняется, как одна из цифр числа.

- Состояние N1. Если при считывании символ равен ‘ ‘, ‘\n’, то происходит перевод числа, составленного из считанных цифр, в десятичный вид и осуществляется его вывод; автомат переходит в состояние А; происходит переприсваивание значений переменным. Если считанный символ != ‘0‘, ‘1’, ‘2‘, ‘3’, ‘4’, ‘ ‘, ‘\n’, то слово является некорректным и автомат переходит в состояние Bad. Если после этого состояние автомата != Bad, а также считанный символ != ‘ ‘, ‘\n’, то считанный символ сохраняется, как одна из цифр числа.

- Состояние Bad. Если при считывании символ равен ‘ ‘, ‘\n’, то автомат переходит в состояние А, при этом если считанный символ равен '\n', то происходит переход на новую строку. Также происходит переприсваивание значений переменным.



**7. Сценарий выполнения работы** [план работы,первоначальный текст программы в черновике(можно на отдельном листе)итесты либо соображения по тестированию].

- Реализовать описанные состояния конечных автоматов и переходов между ними на языке Си.

- Реализовать вывод последнего элемента / числа после того, так считанный символ будет равен EOF.

- Составить ряд тестов, с помощью которых можно было бы проверить корректность работы составленного алгоритма.

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| nan 13 011  56 b10 -40 | 8  -20 |
| sd-13 13  7 8 -sdfs 34227 assa32  -sdf 32 sdfs  sd | 8  17 |
| asdasd asdasda as332  asd212 12 sa312e  q212 1edq  21eq2e1deqw  22w 22 111111 |  |
| 00000013 +77876 -532  f-f-34 + 4 - 3 -ydf +14 | 8  4 3 9 |

**8. Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

#include <stdio.h>

#include <math.h>

enum Value {

A,

N0,

P0,

N1,

P1,

Bad

};

int main(void)

{

int k;

int l;

int x;

long long count\_f;

long long count;

char sym;

k = 1;

count = 0;

enum Value state = A;

while (scanf("%c", &sym) != EOF) {

switch (state) {

case A:

if (sym == ' ' || sym == '\n') {

count = 0;

k = 1;

state = A;

if (sym == '\n') {

printf("\n");

}

}

if (sym == '0' || sym == '1' || sym == '2' || sym == '+') {

state = P0;

k = 1;

}

if (sym == '3' || sym == '4') {

state = P1;

}

if (sym != '0' && sym != '1' && sym != '2' && sym != '3' && sym != '4' && sym != '-' && sym != '+' && sym != ' ' && sym != '\n') {

state = Bad;

}

if (sym == '-' && state != Bad) {

state = N0;

k = -1;

}

if (state != Bad && sym != '-' && sym != '+' && sym != ' ' && sym != '\n') {

count = count \* 10 + (sym - '0');

}

break;

case P0:

if (sym == ' ' || sym == '\n') {

count = 0;

k = 1;

state = A;

if (sym == '\n') {

printf("\n");

}

}

if (sym == '3' || sym == '4') {

state = P1;

}

if (sym != '0' && sym != '1' && sym != '2' && sym != '3' && sym != '4' && sym != ' ' && sym != '\n') {

state = Bad;

}

if (state != Bad && sym != ' ' && sym != '\n') {

count = count \* 10 + (sym - '0');

}

break;

case P1:

if (sym == ' ' || sym == '\n') {

if ((state != Bad) || (count == 0 && sym == '0')) {

l = 0;

count\_f = 0;

while (count >= 1) {

x = count % 10;

x = x \* powf(5, l);

count\_f = count\_f + x;

l = l + 1;

count = count / 10;

}

printf("%lld ", count\_f \* k);

}

k = 1;

state = A;

if (sym == '\n') {

printf("\n");

}

}

if (sym != '0' && sym != '1' && sym != '2' && sym != '3' && sym != '4' && sym != ' ' && sym != '\n') {

state = Bad;

}

if (state != Bad && sym != ' ' && sym != '\n') {

count = count \* 10 + (sym - '0');

}

break;

case N0:

if (sym == ' ' || sym == '\n') {

count = 0;

k = 1;

state = A;

if (sym == '\n') {

printf("\n");

}

}

if (sym == '3' || sym == '4') {

state = N1;

}

if (sym != '0' && sym != '1' && sym != '2' && sym != '3' && sym != '4' && sym != ' ' && sym != '\n') {

state = Bad;

}

if (state != Bad && sym != ' ' && sym != '\n') {

count = count \* 10 + (sym - '0');

}

break;

case N1:

if (sym == ' ' || sym == '\n') {

if ((state != Bad) || (count == 0 && sym == '0')) {

l = 0;

count\_f = 0;

while (count >= 1) {

x = count % 10;

x = x \* powf(5, l);

count\_f = count\_f + x;

l = l + 1;

count = count / 10;

}

printf("%lld ", count\_f \* k);

}

k = 1;

state = A;

if (sym == '\n') {

printf("\n");

}

}

if (sym != '0' && sym != '1' && sym != '2' && sym != '3' && sym != '4' && sym != ' ' && sym != '\n') {

state = Bad;

}

if (state != Bad && sym != ' ' && sym != '\n') {

count = count \* 10 + (sym - '0');

}

break;

case Bad:

if (sym == ' ' || sym == '\n') {

state = A;

count = 0;

k = 1;

}

if (sym == '\n') {

printf("\n");

}

default:

break;

}

}

if (((state == P1 || state == N1) && sym != ' ' && sym != '\n') || (state != Bad && count == 0 && sym == '0')) {

l = 0;

count\_f = 0;

while (count >= 1) {

x = count % 10;

x = x \* powf(5, l);

count\_f = count\_f + x;

l = l + 1;

count = count / 10;

}

printf("%lld\n", count\_f \* k);

}

return 0;

}

**9. Дневник отладки** должен содержать дату и время сеансов отладки и основные события(ошибки в сценарии и программе,

нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
| 1 | дом | 15.11.21 | 14:37 | При пустом вводе программа выдавала в качестве ответа 0 | Учёт предыдущего считанного символа при выводе ответа | Пессимист скажет, что ничего – это ничего. Но чекер – оптимист. Поэтому для него ничего – это ноль. |
| 2 | дом | 15.11.21 | 18:21 | После обработки последовательности программа не переходила на новую строку и чекер не принимал ответ | Изменение условия вывода последнего символа / числа с учетом его корректности | Наверное, чекер создал тот же человек, который написал игру I wanna be the boshy. Для аутентичности не хватает лишь мелодии lol, you died при каждой ошибке на тесте чекера. |
| 3 | дом | 19.11.21 | 18:17 | При вводе строки  Sd-17 14 программа выводила -9, вместо 9 | Переприсваивание k значения 1 после перехода из состояния Bad | \*когда, меняя две строки в коде проходишь чекер\*  “Ого, оно работает!” – удивился, решивший ничему не удивляться, я |

1. **Замечания автора** по существу работы

Работу считаю очень полезной (т.к. через конечные автоматы реализуются многие полноценные программы на Си), интересной и ООООООЧЕНЬ времязатратной. Создавая программы на Си, реализиуемые через конечные автоматы, я обрел много новых знаний в механизме их работы. Также я на практике применил навыки работы с отладчиком gdb, с помощью которого я исправил ряд ошибок в коде программы (10 лаба - one love forever).

1. **Выводы**

Работа мне понравилась в тот момент, когда чекер показал, что моя программа работает корректно. В остальное время я страдал с вылавливанием ошибок в коде. Было не очень занимательно кучу раз переписывать алгоритм, чтобы программу пропустил чекер. Благодаря данной работе я освоил механизмы работы программы, реализуемой с помощью конечных автоматов, что расширило область моих знаний. Информация, полученная мной в ходе выполнения данной лабораторной работы, поможет мне в дальнейшем при составлении более сложных алгоритмов.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_