Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет (институт) Информационных технологий

Кафедра Прикладная математика

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Е.Н. Крючкова

(подпись преподавателя) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Отчет

по расчётному заданию

(название лабораторной работы)

по дисциплине

Теория языков программирования и методы трансляции

(наименование дисциплины)

ЛР 09.03.04.14.000 ОТ

(обозначение документа)

Студент группы ПИ-61 В.Р.Оверченко

(инициалы, фамилия)

Преподаватель профессор, доцент Е.Н. Крючкова

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2020

**Задание**

Программа: главная программа языка С++. Допускается описание функций без параметров, функции возвращают значения допустимых в программе типов.

Типы данных: double, char.

Операции: арифметические и сравнения.

Операторы: присваивания и while.

Операнды: простые переменные, элементы одномерных массивов и константы.

Константы: строковые, символьные и целые в 10 c/c .

**Типы данных:**

|  |  |  |
| --- | --- | --- |
| Тип | Разрядность | Диапазон значений |
| double | 64 разряда | от +/- 1.7E-308 до +/- 1.7E+308 |
| char | 8 разрядов | От -128 до 127 |

**Таблица приведений:**

|  |  |  |  |
| --- | --- | --- | --- |
| Первый тип | Второй тип | Приведение типов | Результат |
| double | double | Нет приведения | double |
| char | double | char приводится к double и вычисляется | double |
| char | char | Нет приведения | double |
| double | char | char приводится к double вычисляется | double |

Целые константы в выражении сразу приводятся к double.

2.3. Перечень операционных символов и связанные с ними семантические подпрограммы

**∆type** **-** определение семантического типа объекта.

**∆identArr** – создание массива (проверка на существование).

**∆identVar** – создание переменной (проверка на существование).

**∆identArr** – создание функции (проверка на существование).

**∆find –** поиск объекта в таблице, возвращение типа данного объекта (при отсутствии - ошибка).

**∆assign** – проверяет соответствие типа выражения с типом идентификатора.

**∆oper -** вычисление типа результата выполнения арифметических и логических действий над операндами в соответствии с таблицей приведения.

**∆unar -** проверка на допустимость выполнения унарной арифметической операции + или – над данными типа.

**∆inc\_dec -** проверка на допустимость выполнения инкрементирования и декрементирования над данными (тип double или целая константа, иначе ошибка).

**∆getTypeConst** – запись константы в таблицу констант.

**∆clean** - очищение верхушки магазина.

**Нисходящая грамматика:**

S --> TS | ε

T --> W | F

W --> double ∆type D; | char ∆type D;

D --> Z D1

D1 --> , Z D1 | ε

Z --> id ∆identVar Z1 | R Z1

Z1 --> = A ∆Assign | ∆clean ε

R --> id [c1 ∆getTypeConst] ∆identArr

F --> double ∆type Y () Q | char ∆type Y () Q

Y --> main ∆identFunc | id ∆identFunc

Q --> {O}

O --> W O | K O | ε

K --> U; | Q | P | H | ;

P --> id ∆find ();

H --> while (A ∆clean) K

U --> id ∆find = A ∆Assign | M | N

M --> ++ G ∆inc\_dec; | -- G ∆inc\_dec; | G M1

M1 --> ++∆inc\_dec; | -- ∆inc\_dec;

N --> G N1

N1 --> += A ∆Assign | -= A ∆Assign | \*= A ∆Assign | /= A ∆Assign | %= A ∆Assign

G --> id ∆find J

J --> [A] | ε

A --> B A1 | + B ∆unar A1 | - B ∆unar A1 |

A1 --> > B ∆Assign A1 | >= B ∆Assign A1 | < B ∆Assign A1 | <= B ∆Assign A1 | == B ∆Assign A1 | A != B ∆Assign A1 | ε

B --> + B1 ∆ oper | - B1 ∆ oper | E

B1 --> + E ∆oper |- E ∆oper

E --> X E1

E1 --> \* X ∆oper E1 | / X ∆oper E1 | % X ∆oper E1 | ε

X --> G | C | (A) |

C --> c1 ∆getTypeConst | c2 ∆getTypeConst | c3 ∆getTypeConst

**Восходящая грамматика:**

S  ST | ε

T  W | F

W  double D; **f(double) = ∆type** | char D; **f(char) = ∆type**

D  D, Z | Z

Z  id | id = A **∆assign** | R = A **∆assign** | R **f(id) = ∆identVar**

R  id [c1] **f(id) = ∆identArr**

F  double Y () Q **f(double) = ∆type** | char Y () Q **f(char) = ∆type**

Y  main **f(main) = ∆identFunc** | id **f(id) = ∆identFunc**

Q  {O} **∆afterSO f( { ) = ∆SO**

O  O W | O K | ε

K  U; | Q | P | H | ;

P  id (); **f(id) =**  **∆find**

H  H (A) K

U  id = A; **f(id) = ∆find** | M | N

M  G ++; **∆inc\_dec (t(G), sign)** | ++ G; **∆inc\_dec (t(G), sign)** | G --; **∆inc\_dec (t(G), sign)** | -- G; **∆inc\_dec (t(G), sign)**

N  G += A; **∆assign (t(G), t(A), sign)** | G -= A; **∆assign (t(G), t(A), sign)** | G \*= A; **∆assign (t(G), t(A), sign)** | G /= A; **∆assign (t(G), t(A), sign)** | G %= A; **∆assign (t(G), t(A), sign)**

G  J | id **f(id) = ∆find**

J  id [A] **f(id) = ∆find**

A → A > B **∆assign (t(A), t(B), sign)** | A >= B **∆assign (t(A), t(B), sign)** | A < B **∆assign (t(A), t(B), sign)** | A <= B **∆assign (t(A), t(B), sign)** | A == B **∆assign (t(A), t(B), sign)** | A != B **∆assign (t(A), t(B), sign)** | + B **t(A) = ∆unar (t(B), sign)** | - B **t(A) = ∆unar (t(B), sign)** | B **t(A) = t(B)**

B → B + E **t(B1) = ∆oper (t(B2), t(E), sign)** | B - E **t(B1) = ∆oper (t(B2), t(E), sign)** | E **t(B1) = t(E)**

E → E \* X **t(E1) = ∆oper(t(E2), t(X), sign)** | E / X **t(E1) = ∆oper(t(E2), t(X), sign)** | E % X **t(E1) = ∆oper(t(E2), t(X), sign)** | X **t(E1) = t(X)**

X → G | C | (A) |

C → c1**∆getTypeConst** | c2 **∆getTypeConst** | c3 **∆getTypeConst**