ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
Институт компьютерных наук и технологий

Отчет о прохождении учебной практики

Сабо Мария Александровна

2 курс, гр. 23531/3
(номер курса обучения и учебной группы)

09.03.01 «Информатика и вычислительная техника»
(Направление подготовки (код и наименование)

Место прохождения практики: кафедра КСПТ ИКНТ
(указывается наименование профильной организации или наименование структурного подразделения ФГАОУ ВО «СПбПУ»

Сроки практики: 25 июня по 21 июля 2018 г.

Руководитель практики : Сидорина Татьяна Леонидовна

Оценка:

Руководитель практики /Сидорина Т.Л

Обучающийся: /Сабо М.А

Дата:

СОДЕРЖАНИЕ

Введение

В качестве ознакомительной практики я взяла написание моего учебного проекта “Интерпретатор языка LOOP” на объектно-ориентированном языке программирования C++. Тем самым, код становится более читаемым.

УТВЕРЖДАЮ
Зав. кафедрой КСПТ ИКНТ
_____ В.М. Ицыксон
подпись

«___» _____ 2018 г.

ИНДИВИДУАЛЬНЫЙ ПЛАН УЧЕБНОЙ ПРАКТИКИ

(вид и тип практики)

Ф.И.О. Обучающегося Сабо Мария Александровна
Учебное подразделение: кафедра Компьютерных систем и программных технологий ИКНТ.
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Руководитель практики от ФГАОУ ВО «СПбПУ»:
_____

Сроки проведения: _25 июня по 21 июля 2018 г._____

Содержание практики: Разработка интерпретатора языка LOOP.

Форма итоговой отчетности по практике:___зачет_____
Перечень учебной литературы и методических материалов, в том числе ресурсы сети «Интернет»
Материально-техническая база _____
_____

Обучающийся _____ Сабо М.А

Руководитель практики _____ Сидорина Т.Л

1. Техническое задание

Входные параметры программы:

- имя файла с текстом программы (формат .txt) (Таблица 1)
- имя файла (формат .txt), где переменные перечислены в виде: x1=3, x2=5, x10=6 и тд (Таблица 2)
- имя выходного файла с результатом работы программы (формат .txt) (Таблица 3)

Командная строка : niloop.exe program.txt parameters.txt output.txt COMLIMIT(not necessary)

Default: COMLIMIT = 100

Выходные данные программы:

Программа формирует текстовый файл с результатом синтаксического разбора и результатами работы программы.

В файле содержится следующая информация:

- переданная команда
- результат синтаксического разбора (Таблица 3)
- значения переменных после выполнения команды
- в последней строке значение x0

Таблица 1.

```
program.txt
x1 := x2 + 3;
x1 := 6;
x0 := x1 + 1;
LOOP x1
DO
x6 := x1 + 2;
x5 := x6 + 4;
END;
x6 = x5 + 1;
```

Таблица 3.

```
output.txt
x1:=x2 + 3;
x1:=6;
x0:=x1+1;
LOOP x1
DO
x6:=x1+2;
x5:=x6+4;
END;
x6=x5+1;
Нет синтаксических ошибок.
x1 := x2 + 3;     x1 = 4 x2 = 1 x3 = 8
x1 := 6;          x1 = 6 x2 = 1 x3 = 8
x0 := x1 + 1;     x0 = 7 x1 = 6 x2 = 1 x3 = 8
DO                x0 = 7 x1 = 6 x2 = 1 x3 = 8
x6 := x1 + 2;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x6 = 8
x5 := x6 + 4;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x6 := x1 + 2;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x5 := x6 + 4;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x6 := x1 + 2;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x5 := x6 + 4;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x6 := x1 + 2;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x5 := x6 + 4;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x6 := x1 + 2;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x5 := x6 + 4;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x6 := x1 + 2;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x5 := x6 + 4;     x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
END;              x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 8
x6 = x5 + 1;      x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 13
x0 = 7            x0 = 7 x1 = 6 x2 = 1 x3 = 8 x5 = 12 x6 = 13
```

Таблица 2.

```
parametres.txt
x2 = 1, x3 = 8
```

Программа выводит результат синтаксического разбора на экран и в файл. Если результат отрицательный, то программа не начинает работать.

2. Метод решения

Программа запускается командной строкой:
niloop.exe <имя файла программы> <имя файла со значениями переменных> <имя выходного файла> [количество выполняемых команд]
Если параметров передано меньше, программа выводит справку.
Если с одним параметром help – выводится более подробная справка.
При работе программы происходит загрузка текста программы из файла программы, загрузка текста из файла с переменными, синтаксический разбор этих файлов и запуск программы на выполнение. Результат выполнения программы записывается в выходной файл. Если выходной файл не открыть, то программа показывает результат выполнения только на экран.
Программа состоит из следующих модулей:

-**niloop.cpp** – главный модуль программы.

В модуле производится разбор параметров, открытие входных и выходного файлов, запуск загрузки текстов программы и параметров в списки класса Iloop, вызов процедуры синтаксического разбора, запуск программы на исполнение.

-**load.cpp** – программа загрузки текстов файлов.

В модуле из файлов с текстом программы и инициализации переменных считывается информация в элементы типа Command класса Iloop – iloop.prog_list, iloop.var_list.

-**checksyntax.cpp** – программа синтаксического разбора текста.

По спискам класса Iloop – iloop.prog_list, iloop.var_list, содержащим элементы типа Command. В результате работы программы в элемент Command записывается номер команды, номер синтаксической ошибки, значения индексов переменных и константы.

Номера команд:

1 – DO
2 – END
3 – LOOPx
4 – :=+ (присвоение с плюсом)
5 – :=- (привоение с минусом)
6 – = (инициализация переменной)

-**doProg.cpp** – программа, выполняющая загруженную программу.

По спискам класса Iloop – iloop.prog_list, iloop.var_list, содержащим элементы типа Command, с помощью getvar(), putvar(), getloop(), putloop(), checkloop(), findvar() формирует списки varEl_list (из элементов типа VarEl), loopEl_list (из элементов типа LoopEl).

-Command.cpp – функции элемента типа Command, в т.ч описание функции, которая переводит номера ошибок в текстовое представление.
-Iloop.cpp – содержит фунуции печати списков класса Iloop – loopEl_list, varEl_list, prog_list, var_list.
-VarEl.cpp – содержит функцию печати элемента .
-LoopEl.cpp – содержит функцию печати элемента .
-VarEl.h – описание класса varEl, содержащего поля index, value и функции работы с нею.
-LoopEl.h – описание класса loopEl, содержащего поля count, index_command.
-Iloop.h – описание класса Iloop .
-Command.h – описание класса Command .

3. Коды ошибок

Код ошибки	Описание ошибки	Действие
1	Garbage characters in the use of DO	Завершение программы
2	Garbage characters in the use of END	Завершение программы
3	Garbage characters in the use of LOOP-X	Завершение программы
4	Garbage characters in the index of LOOP x	Завершение программы
5	Garbage characters in the left part of +-	Завершение программы
6	Garbage characters in the left part of :=	Завершение программы
7	Garbage characters in the index of left part	Завершение программы
8	Plus or minus are not found	Завершение программы
9	There is unrecognised command	Завершение программы
10	Garbage characters in the use of x	Завершение программы
11	Garbage characters in the index of x	Завершение программы
12	Garbage characters in the right part of =	Завершение программы
13	There is unrecognised command	Завершение программы
14	Count of DO greater than LOOP	Завершение программы
15	Count of END greater than LOOP or DO	Завершение программы
21	The x is not available	Завершение программы
22	The x is not initialised	Завершение программы
23	Garbage characters in the right part of +- \0	Завершение программы
24	Limit of index is exceeded	Завершение программы

Ошибки загрузки программы:

Описание ошибки	Действие
Command is too long	Завершение программы
String is too long	Завершение программы

Ошибки при передаче параметров:

Описание ошибки	Действие
Input file name and output file name couldn't equal	Завершение работы
Wrong COMLIMIT paramrter	Завершение работы
Output file can not be create	Завершение работы

4. Программа и методы испытаний

4.1 Запуск программы без аргументов или с параметром help.

```
C:\Users\808625\source\repos\niloop\Debug>niloop.exe
Author: Sabo Maria
        This program interpret programs written in the LOOP language.
        For access the commands used, run the program with the key. 'help'
        For running the program you need to write paths to files.
        Unnecessary COMLIMIT -  positive integer. Default COMLIMIT = 100.
        Example: iloop.exe program.txt var.txt out.txt [COMLIMIT](not necessary)

Для продолжения нажмите любую клавишу . . .

C:\Users\808625\source\repos\niloop\Debug>niloop.exe help
        For access the commands used, run the program with the key. 'help'
        For running the program you need to write paths to files.
        Unnecessary COMLIMIT -  positive integer. Default COMLIMIT = 100.
        Example: iloop.exe program.txt var.txt out.txt [COMLIMIT] (not necessary)

You can use commands such as this:
        xN := xM +- C;
        LOOP xi
        DO
        something...
        END;
You can input parametres such as this: x0 = C1, ... xN = CN
You can use only non-negative integer
```

4.2 Запуск программы с несуществующим входным файлом.

```
c:\temp_practice>niloop.exe notexistfile.txt var6.txt out.txt
File notexistfile.txt is not found
Для продолжения нажмите любую клавишу . . .
```

4.3 Запуск программы с несуществующим входным файлом переменных.

```
c:\temp_practice>niloop.exe prog1.txt notexistfile.txt out.txt
File notexistfile.txt is not found
Для продолжения нажмите любую клавишу . . .
```

4.4 Синтаксические ошибки программы.

```
c:\temp_practice>niloop.exe prog2.txt var6.txt out.txt
0 x1=6
1 x2=6
2 x0=4
3 x3=5
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x2:=x1-czzcx7  er: 23 Garbage characters in the right part of +-
1 xzcx3:=xxvcxxcv2+3   er: 7 Garbage characters in the index of left part
2 LOOcxvcxvPx1 er: 13 There is unrecognised command
3 DO  er: 14 Count of DO greater than LOOP
4 LOOPx1
5 DO  er: 14 Count of DO greater than LOOP
6 x0:=xczzxcxzzc0+1    er: 5 Garbage characters in the left part of +-
7 END
8 x4:=x2-4
9 END  er: 15 Count of END greater than LOOP or DO
Programm: 1 LOOP - 2 DO - 2 END error found
Programm: 1 syntax error found
Для продолжения нажмите любую клавишу . . .
The programm is completed
Для продолжения нажмите любую клавишу . . .
```

4.5 Ошибки выполнения программы.

Используемая переменная не инициализирована.

```
c:\temp_practice>niloop.exe prog1.txt var7.txt out.txt
0 x2=6
1 x0=4
2 x3=5
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x2:=x1-7
1 x3:=x2+3
2 LOOPx1
3 DO
4 LOOPx1
5 DO
6 x0:=x0+1
7 END
8 x4:=x2-4
9 END
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x2=6
                x2 = 6;
1 x0=4
                x2 = 6;  x0 = 4;
2 x3=5
                x2 = 6;  x0 = 4;  x3 = 5;
0 x2:=x1-7  er: 22 The x is not initialised
                x2 = 6;  x0 = 4;  x3 = 5;
The program is interrupted
x0 = 4
Для продолжения нажмите любую клавишу . . .
```

4.6 Пример успешного выполнения программы.

Входные файлы:

prog1.txt — Блокнот

Файл Правка Формат Вид Справка

```
x2:=x1-7;
x3:=x2+3;

LOOP x1
DO
        LOOP x1
        DO
                x0:=x0+1;
        END;
        x4:=x2-4;
END;
```

var8.txt — Блокнот

Файл Правка Формат Вид Справка

```
x1 = 1, x2 = 1, x0 = 1, x3 = 1
```

Результат работы программы:

```
c:\temp_practice>niloop.exe prog1.txt var8.txt out.txt
0 x1=1
1 x2=1
2 x0=1
3 x3=1
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x2:=x1-7
1 x3:=x2+3
2 LOOPx1
3 DO
4 LOOPx1
5 DO
6 x0:=x0+1
7 END
8 x4:=x2-4
9 END
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x1=1
                x1 = 1;
1 x2=1
                x1 = 1;  x2 = 1;
2 x0=1
                x1 = 1;  x2 = 1;  x0 = 1;
3 x3=1
                x1 = 1;  x2 = 1;  x0 = 1;  x3 = 1;
0 x2:=x1-7
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = 1;
1 x3:=x2+3
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;
2 LOOPx1
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;
3 DO
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;
4 LOOPx1
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;  1 from 4;
5 DO
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;  1 from 4;
6 x0:=x0+1
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;      1 from 2;  1 from 4;
7 END
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;      1 from 2;
8 x4:=x2-4
```

```
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;  x4 = -10;      1 from 2;
9 END
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;  x4 = -10;
x0 = 2
Для продолжения нажмите любую клавишу . . .
The programm is completed
Для продолжения нажмите любую клавишу . . .
```

Результат работы программы, записанный в выходной файл:

out.txt — Блокнот

Файл Правка Формат Вид Справка

```
0 x1=1
1 x2=1
2 x0=1
3 x3=1
0 x2:=x1-7
1 x3:=x2+3
2 LOOPx1
3 DO
4 LOOPx1
5 DO
6 x0:=x0+1
7 END
8 x4:=x2-4
9 END
0 x1=1
                x1 = 1;
1 x2=1
                x1 = 1;  x2 = 1;
2 x0=1
                x1 = 1;  x2 = 1;  x0 = 1;
3 x3=1
                x1 = 1;  x2 = 1;  x0 = 1;  x3 = 1;
0 x2:=x1-7
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = 1;
1 x3:=x2+3
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;
2 LOOPx1
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;
3 DO
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;
4 LOOPx1
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;  1 from 4;
5 DO
                x1 = 1;  x2 = -6;  x0 = 1;  x3 = -3;      1 from 2;  1 from 4;
6 x0:=x0+1
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;      1 from 2;  1 from 4;
7 END
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;      1 from 2;
8 x4:=x2-4
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;  x4 = -10;      1 from 2;
9 END
                x1 = 1;  x2 = -6;  x0 = 2;  x3 = -3;  x4 = -10;
```

4.7 Пример программы, которая требует большого количества выполнения команд.

prog1.txt — Блокнот

Файл Правка Формат Вид Спра

```
x2:=x1-7;
x3:=x2+3;

LOOP x1
DO
        LOOP x1
        DO
                x0:=x0+1;
        END;
        x4:=x2-4;
END;
```

var6.txt — Блокнот

Файл Правка Формат Вид Справка

```
x1 = 6, x2 = 6, x0 = 4, x3 = 5
```

```
c:\temp_practice>niloop.exe prog1.txt var6.txt out.txt
0 x1=6
1 x2=6
2 x0=4
3 x3=5
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x2:=x1-7
1 x3:=x2+3
2 LOOPx1
3 DO
4 LOOPx1
5 DO
6 x0:=x0+1
7 END
8 x4:=x2-4
9 END
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x1=6
                x1 = 6;
1 x2=6
                x1 = 6;  x2 = 6;
2 x0=4
                x1 = 6;  x2 = 6;  x0 = 4;
3 x3=5
                x1 = 6;  x2 = 6;  x0 = 4;  x3 = 5;
0 x2:=x1-7
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 5;
1 x3:=x2+3
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;
2 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;      6 from 2;
3 DO
```

Далее спрашивается, выводить еще 100 команд или нет.

```
                x1 = 6;  x2 = -1;  x0 = 28;  x3 = 2;  x4 = -5;      2 from 2;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 28;  x3 = 2;  x4 = -5;      2 from 2;  6 from 4;
5 DO
                x1 = 6;  x2 = -1;  x0 = 28;  x3 = 2;  x4 = -5;      2 from 2;  6 from 4;
6 x0:=x0+1
                x1 = 6;  x2 = -1;  x0 = 29;  x3 = 2;  x4 = -5;      2 from 2;  6 from 4;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 29;  x3 = 2;  x4 = -5;      2 from 2;  5 from 4;
5 DO
                x1 = 6;  x2 = -1;  x0 = 29;  x3 = 2;  x4 = -5;      2 from 2;  5 from 4;
6 x0:=x0+1
                x1 = 6;  x2 = -1;  x0 = 30;  x3 = 2;  x4 = -5;      2 from 2;  5 from 4;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 30;  x3 = 2;  x4 = -5;      2 from 2;  4 from 4;
5 DO
                x1 = 6;  x2 = -1;  x0 = 30;  x3 = 2;  x4 = -5;      2 from 2;  4 from 4;

100commands completed
0 - continue
1 - exit
```

```
6 x0:=x0+1
                x1 = 6;  x2 = -1;  x0 = 38;  x3 = 2;  x4 = -5;      1 from 2;  3 from 4;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 38;  x3 = 2;  x4 = -5;      1 from 2;  2 from 4;
5 DO
                x1 = 6;  x2 = -1;  x0 = 38;  x3 = 2;  x4 = -5;      1 from 2;  2 from 4;
6 x0:=x0+1
                x1 = 6;  x2 = -1;  x0 = 39;  x3 = 2;  x4 = -5;      1 from 2;  2 from 4;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 39;  x3 = 2;  x4 = -5;      1 from 2;  1 from 4;
5 DO
                x1 = 6;  x2 = -1;  x0 = 39;  x3 = 2;  x4 = -5;      1 from 2;  1 from 4;
6 x0:=x0+1
                x1 = 6;  x2 = -1;  x0 = 40;  x3 = 2;  x4 = -5;      1 from 2;  1 from 4;
7 END
                x1 = 6;  x2 = -1;  x0 = 40;  x3 = 2;  x4 = -5;      1 from 2;
8 x4:=x2-4
                x1 = 6;  x2 = -1;  x0 = 40;  x3 = 2;  x4 = -5;      1 from 2;
9 END
                x1 = 6;  x2 = -1;  x0 = 40;  x3 = 2;  x4 = -5;
x0 = 40
Для продолжения нажмите любую клавишу . . .
The programm is completed
Для продолжения нажмите любую клавишу . . .
```

Также можем запустить с параметром COMLIMIT.

```
c:\temp_practice>niloop.exe prog1.txt var6.txt out.txt 200
0 x1=6
1 x2=6
2 x0=4
3 x3=5
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x2:=x1-7
1 x3:=x2+3
2 LOOPx1
3 DO
4 LOOPx1
5 DO
6 x0:=x0+1
7 END
8 x4:=x2-4
9 END
Programm: 0 syntax error found
Для продолжения нажмите любую клавишу . . .
0 x1=6
                x1 = 6;
1 x2=6
                x1 = 6;  x2 = 6;
2 x0=4
                x1 = 6;  x2 = 6;  x0 = 4;
3 x3=5
                x1 = 6;  x2 = 6;  x0 = 4;  x3 = 5;
0 x2:=x1-7
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 5;
1 x3:=x2+3
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;
2 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;      6 from 2;
3 DO
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;      6 from 2;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;      6 from 2;  6 from 4;
5 DO
                x1 = 6;  x2 = -1;  x0 = 4;  x3 = 2;      6 from 2;  6 from 4;
6 x0:=x0+1
                x1 = 6;  x2 = -1;  x0 = 5;  x3 = 2;      6 from 2;  6 from 4;
4 LOOPx1
                x1 = 6;  x2 = -1;  x0 = 5;  x3 = 2;      6 from 2;  5 from 4;
```

Программа выполняется без прерываний.

5. Заключение

Таким образом, вследствие разработки проекта “Интерпретатор LOOP” были изучены основы языка C++.

6. Список использованных источников

Основы программирования на языках Си и C++ для начинающих[Электронный ресурс]
Режим доступа: http://cppstudio.com/

Справочник по языку C++
Режим доступа: https://msdn.microsoft.com/ru-ru/library/3bstk3k5.aspx

7. ПРИЛОЖЕНИЕ

7.1 Приложение 1 “niloop.cpp”

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "Iloop.h"

#define COMLIMIT 100

using namespace std;

int main(int argc, char* argv[])
{
	setlocale(LC_ALL, "rus");
	ifstream file_prog;
	ifstream file_var;
	ofstream ofile;

	// вывод справки
	if (argc == 2 && strcmp(argv[1], "help") == 0) {
		cout  <<
			"\tFor access the commands used, run the program with the key. 'help'
\n"
			"\tFor running the program you need to write paths to files. \n"
			"\tUnnecessary COMLIMIT -  positive integer. Default COMLIMIT = 100.
\n"
			"\tExample: iloop.exe program.txt var.txt out.txt [COMLIMIT] (not
necessary)\n\n"
			"You can use commands such as this: \n\txN := xM +- C;\n\tLOOP
xi\n\tDO\n\tsomething...\n\tEND;\n"
			"You can input parametres such as this: x0 = C1, ... xN = CN \n"
			"You can use only non-negative integer \n" << endl;
		system("pause");
		return 0;
	}
	if (argc < 4) {
		cout << "Author: Sabo Maria \n"
			"\tThis program interpret programs written in the LOOP language.\n"
			"\tFor access the commands used, run the program with the key. 'help'
\n"
			"\tFor running the program you need to write paths to files. \n"
			"\tUnnecessary COMLIMIT -  positive integer. Default COMLIMIT = 100.
\n"
			"\tExample: iloop.exe program.txt var.txt out.txt [COMLIMIT](not
necessary) \n" << endl;
		system("pause");
		return 0;
	}
	if (strcmp(argv[1], argv[3]) == 0 || strcmp(argv[2], argv[3]) == 0) {
		cout << "Input file name and output file name couldn't equal" << endl;
		return 0;
	}
	// необязательный пятый аргумент
	int comlimit;
	if (argc == 5) {
		comlimit = atoi(argv[4]);
		if (comlimit <= 0) {
			cout << "Wrong COMLIMIT parameter \n" << endl;
			return 0;
		}
```

```
	}
	else
		comlimit = COMLIMIT;


	ofile.open(argv[3]);

	if (!ofile.is_open()) {
		cout << "Output file " << argv[3] << " can not be create" << endl;
		system("pause");
		return 0;
	}
	file_prog.open(argv[1]);
	if (!file_prog.is_open()) {
		cout << "File " << argv[1] << " is not found" << endl;
		system("pause");
		return 0;
	}
	else {
		file_var.open(argv[2]);
		if (!file_var.is_open()) {
			cout << "File " << argv[2] << " is not found" << endl;
			system("pause");
			return 0;
		}
		else {
			// выполнение программы, если файлы загружены и синтаксический разбор
не имеет ошибок
			Iloop iloop;
			if (iloop.loadListProg(file_prog) && iloop.loadListVar(file_var)) {
				if (iloop.checkSyntax(ofile))
					iloop.doProg(ofile, comlimit);
			}
			else
				cout << "Load program error" << endl;
			file_var.close();
		}
		file_prog.close();
	}
	if (ofile.is_open()) ofile.close();
	cout << "The programm is completed" << endl;
	system("pause");
	return 0;

}
```

7.2 Приложение 2 “load.cpp”

```
#include "stdafx.h"
#include "Iloop.h"

int Iloop::loadListProg(ifstream& F) {
	if (F.is_open()) {
		while (!F.eof())	{
			string buffer = "";
			string buffer1  = "";
			string buffer2  = "";
			getline(F, buffer,';');
		    buffer.erase (remove(buffer.begin(), buffer.end(), ' '), buffer.end());
			buffer.erase (remove(buffer.begin(), buffer.end(), '\n'),
buffer.end());
			buffer.erase (remove(buffer.begin(), buffer.end(), '\t'),
buffer.end());
			if (buffer != "") {
			    int ex = 1;
```

```
                        while(ex) {
                                size_t pos = buffer.find("DO");
                                if (pos == string::npos){
                                        Command com(0, buffer, 0, 0, 0, 0, 0);
                                        prog_list.push_back(com);
                                        ex = 0;
                                }
                                else {
                                        buffer1 = buffer.substr(0,pos);
                                        Command com(0, buffer1, 0, 0, 0, 0, 0);
                                        prog_list.push_back(com);
                                        Command com1(0, "DO", 0, 0, 0, 0, 0);
                                        prog_list.push_back(com1);
                                        buffer2 = buffer.substr(pos+2);
                                        buffer = buffer2;
                                }
                        }
                }
        }
        //printList(prog_list, F);
                return 1;
        }
        else
                return 0;
}
int Iloop::loadListVar(ifstream& F) {
        if (F.is_open()) {
                while (!F.eof())    {
                        string buffer = "";
                        getline(F, buffer,',');
                    buffer.erase (remove(buffer.begin(), buffer.end(), ' '), buffer.end());
                        buffer.erase (remove(buffer.begin(), buffer.end(), '\n'),
buffer.end());
                        buffer.erase (remove(buffer.begin(), buffer.end(), '\t'),
buffer.end());
                                Command com(0, buffer, 0, 0, 0, 0, 0);
                                var_list.push_back(com);

                }
//              printList(var_list, NULL);
                return 1;
        }
        else
                return 0;
        }
```

7.3 Приложение 3 “checkSyntax.cpp”

```
#include "stdafx.h"
#include <iostream>
#include "Iloop.h"

int Iloop::checkSyntax(ofstream& F) {
        int k_end = 0;
        int k_do = 0;
        int k_loop = 0; // количество end, do, loop в программе      int er = 0;
        int er = 0;
        int nc = 0;
        int errvar_count = 0;
        int errprog_count = 0;

        for (int i = 0; i < var_list.size(); i++) {
                var_list[i] = checkvline(var_list[i].get_commandText(), i);
                if (var_list[i].get_error() != 0)
                        errvar_count++;
```

```
	}
	printList(var_list, F);
	cout << "Programm: " << errvar_count << " syntax error found\n";
	system("pause");

	er = 0;
	nc = 0;
	for (int i = 0; i < prog_list.size(); i++) {
		prog_list[i] = checkpline(prog_list[i].get_commandText(), i);
		if (prog_list[i].get_error() != 0)
			errvar_count++;
		switch (prog_list[i].get_command()) {
		case 1:
			k_loop++;
			break;
		case 2:
			k_do++;
			if (k_do > k_loop)
				prog_list[i].put_error(14);
			break;
		case 3:
			k_end++;
			if (k_end > k_loop || k_end > k_do)
				prog_list[i].put_error(15);
			break;
		}
	}
	printList(prog_list, F);
	if (k_end != k_loop || k_do != k_loop || k_end != k_do) {
		cout <<"Programm: " << k_loop << " LOOP - " << k_do << " DO - " << k_end <<
" END error found\n";
		errprog_count++;
	}
	cout << "Programm: " << errprog_count << " syntax error found\n";
	system("pause");
	if (errvar_count + errprog_count != 0)
		return 0;
	else
		return 1;

}

Command Iloop::checkpline(string text, int ns) {
	int of, p1, p2, p3;
	Command com(ns, text, 0, 0, 0, 0, 0);

	size_t pos = text.find("DO");
	if (pos != string::npos) {
		// DO входит в строку
		if (text.length() == 2) {
			// 2 - длина DO
			com.set_command(ns, text, 2, 0, 0, 0, 0);
		}
		else
			com.set_command(ns, text, 2, 1, 0, 0, 0);
	}
	else {
		size_t pos = text.find("END");
		if (pos != string::npos) {
			// END входит в строку
			if (text.length() == 3) {
				// 2 - длина END
					com.set_command(ns, text, 3, 0, 0, 0, 0);
			}
			else
```

```
                        com.set_command(ns, text, 3, 2, 0, 0, 0);
                }
                else {
                        size_t pos = text.find("LOOPx");
                        if (pos != string::npos) {
                                // LOOPx входит в строку
                                if (text.length() == 5 || pos != 0) {
                                        // 5 - длина LOOPx
                                        com.set_command(ns, text, 1, 3, 0, 0, 0);
                                }
                                else {
                                                of = 1;
                                                for (size_t i = 5; i < text.length(); i++) {
                                                        if (text[i] < 48 || text[i] > 57)  //
после x стоит число, состоящее не из цифр (проверка по ASCII)
                                                                of = 0;
                                                }
                                                if (!of)
                                                        com.set_command(ns, text, 1, 4, 0, 0, 0);
// ошибка в индексе x после LOOP
                                                else {
                                                        p1 = atoi(text.substr(5).c_str()); // в p1
будет храниться индекс x
                                                        if (p1 <= INT_MAX && p1 >= 0) {
                                                                com.set_command(ns, text, 1, 0, p1,
0, 0);
                                                        }
                                                        else
                                                                com.set_command(ns, text, 1, 24, 0,
0, 0);
                                                }
                                }
                        }
                        else {
                                // присваивание
                                size_t pos = text.find(":=");
                                if (pos != string::npos) {
                                        string left = text.substr(0, pos);
                                        string right = text.substr(pos+2);
                                        size_t pos = left.find("x");
                                        if (pos != string::npos) {
                                                if (pos != 0 || left.length() == 1) {
                                                        com.set_command(ns, text, 4, 6, 0, 0, 0);
// мусорв левой части
                                                }
                                                else {
                                                        of = 1;
                                                        for (size_t i = 1; i < left.length(); i++)
{
                                                                if ((int)left[i] < 48 ||
(int)left[i] > 57) { // после x не стоит число, состоящее из цифр (проверка по ASCII)
                                                                        of = 0;
                                                                }
                                                        }
                                                        if (!of) {
                                                                com.set_command(ns, text, 4, 7, 0,
0, 0);
                                                        }
                                                        else {
                                                                p1 = atoi(left.substr(1).c_str());
// в p1 будет храниться индекс x
                                                                if (p1 > INT_MAX || p1 < 0) {
                                                                        com.set_command(ns, text, 4,
24, 0, 0, 0);
                                                                }
```

```
                                                        else {
                                                               size_t pos =
right.find("+");
                                                               if (pos != string::npos) {
                                                                      //+
                                                                      string rleft =
right.substr(0, pos);
                                                                      string rright =
right.substr(pos + 1);
                                                                      size_t pos =
rleft.find("x");
                                                                      if (pos !=
string::npos) {
                                                                             //
                                                                             if (pos != 0 ||
rleft.length() == 1) {
       com.set_command(ns, text, 5, 5, 0, 0, 0);
                                                                      }
                                                                      else {
                                                                             of = 1;
                                                                             for
(size_t i = 1; i < rleft.length(); i++)
       if (rleft[i] < 48 || rleft[i] > 57) // после x не стоит число, состоящее из цифр
(проверка по ASCII)
       of = 0;
                                                                             if (!of)
{
       com.set_command(ns, text, 4, 5, 0, 0, 0);
                                                                             }
                                                                             else {
       p2 = atoi(rleft.substr(1).c_str()); // в p2 будет храниться индекс xM
       if (p2 > INT_MAX || p2 < 0) {
       com.set_command(ns, text, 4, 24, 0, 0, 0);
                                                                                    }
       else {
       of = 1;
       for (int i = 0; i < rright.length(); i++) {
              if ((int)rright[i] < 48 || (int)rright[i] > 57)
                     of = 0;
       }
       if (!of) {
              com.set_command(ns, text, 4, 23, 0, 0, 0);
       }
       else {
              p3 = atoi(rright.c_str()); // в p3 будет храниться C
              com.set_command(ns, text, 4, 0, p1, p2, p3);
```

```
}
}
}
}
}
else {
com.set_command(ns, text, 4, 5, 0, 0, 0);
}
}
else {
size_t pos =
right.find("-");
if (pos !=
string::npos) {
//-
string rleft =
right.substr(0, pos);
string rright =
right.substr(pos + 1);
size_t pos =
rleft.find("x");
if (pos !=
string::npos) {
//
if (pos
!= 0 || rleft.length() == 1) {
com.set_command(ns, text, 5, 5, 0, 0, 0);
}
else {
of = 1;
for (size_t i = 1; i < rleft.length(); i++)
if (rleft[i] < 48 || rleft[i] > 57) // после x не стоит число, состоящее из цифр
(проверка по ASCII)
of = 0;
if (!of) {
com.set_command(ns, text, 5, 5, 0, 0, 0);
}
else {
p2 = atoi(rleft.substr(1).c_str()); // в p2 будет храниться индекс xM
if (p2 > INT_MAX || p2 < 0) {
com.set_command(ns, text, 5, 24, 0, 0, 0);
}
else {
of = 1;
for (int i = 0; i < rright.length(); i++) {
if ((int)rright[i] < 48 || (int)rright[i] > 57)
```

```
                        of = 0;
                }
                if (!of) {
                        com.set_command(ns, text, 5, 23, 0, 0, 0);
                }
                else {
                        p3 = atoi(rright.c_str()); // в p3 будет храниться C
                        com.set_command(ns, text, 5, 0, p1, p2, p3);
                }
        }
                                                                                }
                                                                        }
                                                                }
                                                                else {
        com.set_command(ns, text, 5, 5, 0, 0, 0);
                                                                }
                                                        }
                                                        else {
        com.set_command(ns, text, 4, 8, 0, 0, 0); // no +-
                                                        }
                                                }
                                        }
                                }
                        }
                        else {
                                com.set_command(ns, text, 4, 6, 0, 0, 0); //
мусор в левой части
                        }
                }
                else {
                        com.set_command(ns, text, 7, 13, 0, 0, 0); //
неизвестная команда
                }
        }
    }
    return com;
}
Command Iloop::checkvline(string text, int ns) {
    int of, p1, p2;
    Command com(ns, text, 0, 0, 0, 0, 0);
    size_t pos = text.find("=");
    if (pos == string::npos) {
        com.set_command(ns, text, 7, 13, 0, 0, 0);
    }
    else {
        string left = text.substr(0, pos);
        string right = text.substr(pos+1);
        pos = left.find("x");
        if (pos == string::npos)
```

```
                com.set_command(ns, text, 6, 10, 0, 0, 0);
            else {
                if (pos !=0 || left.length() == 1)
                    com.set_command(ns, text, 6, 10, 0, 0, 0);
                else {
                    of = 1;
                    for (int i = 1; i < left.length(); i++) {
                        if (left[i] < 48 || left[i] > 57)  // после x не стоит
число, состоящее из цифр (проверка по ASCII)
                            of = 0;
                    }
                    if (!of)
                        com.set_command(ns, text, 6, 11, 0, 0, 0); // после x
не цифровые символы
                    else {
                        p1 = atoi(left.substr(1).c_str());
                        if (p1 > INT_MAX || p1 < 0)
                            com.set_command(ns, text, 6, 24, p1, 0, 0); //
индекс превышен
                        else {
                            of = 1;
                            for (int i = 1; i < right.length(); i++) {
                                if (right[i] < 48 || right[i] > 57) { //
после x не стоит число, состоящее из цифр (проверка по ASCII)
                                    of = 0;
                                }
                            }
                            if (!of) {
                                com.set_command(ns, text, 6, 12, p1, 0,
0); // некорректная записать в правой части
                            }
                            else {
                                p2 = atoi(right.c_str());
                                com.set_command(ns, text, 6, 0, p1, p2,
p2);
                            }
                        }
                    }
                }
            }
        }
        return com;
}
```

7.4 Приложение 4 “doProg.cpp”

```
#include "stdafx.h"
#include "Iloop.h"

void Iloop::doProg(ofstream& F, int comlimit) {

        int er;
        for (int i = 0; i < var_list.size(); i++) {
            er = doCommand(var_list[i].get_command(), var_list[i].get_p1(),
var_list[i].get_p2(), var_list[i].get_p3(), &i);
            var_list[i].put_error(er);
            // печать команды
            cout << var_list[i].toString() << endl;
            cout << "                " << varToString() << "    " << loopToString() <<
endl;
            F << var_list[i].toString() << endl;
            F << "                " << varToString() << "    " << loopToString() <<
endl;
```

```
			if (er != 0)
				break;
		}
		if (er != 0) cout << "Variable initialisation error" << endl;

		int cnt_temp = 0;
		int cnt_all = 0;
		int decision = 0; // решение продолжить ли работу программы
		for (int i = 0; i < prog_list.size(); i++) {
			er = doCommand(prog_list[i].get_command(), prog_list[i].get_p1(),
prog_list[i].get_p2(), prog_list[i].get_p3(), &i);
			if (er != -1) {
				prog_list[i].put_error(er);
				// печать команды
				cout << prog_list[i].toString() << endl;
				cout << "                " << varToString() << "    " <<
loopToString() << endl;
				F << prog_list[i].toString() << endl;
				F << "                " << varToString() << "    " << loopToString()
<< endl;
			}
			if (er != -1 && er != 0)
				break;
			cnt_temp++;
			if (cnt_temp == comlimit) {
				cnt_all = cnt_all + cnt_temp;
				cnt_temp = 0;
				cout << endl << cnt_all << "commands completed \n0 - continue \n1 -
exit" << endl;

				scanf("%d", &decision);
				if (decision == 1) {
					cout << "The program is interrupted by user" << endl;
					break;
				}
			}
		}
		if (er != -1 && er != 0) cout << "The program is interrupted" << endl;
		if (findvar(0)) {
			cout << "x0 = " << getvar(0) << endl;
		}
		else {
			cout << "x0 is not initialised" << endl;
		}
		system("pause");
}


int  Iloop::doCommand(int c, int p1, int p2, int p3, int * index) {

		if (!checkloop() && c != 2) // выход, если последний loop нулевой и не END
			return -1;
		int ok = 0;
		int val;
		switch (c) {
		case 1: // LOOPx
			if (!findvar(p1))
				ok = 22;
			else
				if (!putloop(p1, *index))
					ok = 21;
			break;
		case 2: // DO
			break;
```

```
        case 3: // END
                *index = getloop(index);
                break;
        case 4: // :=+
                if (!findvar(p2))
                        ok = 22;
                else {
                        val = getvar(p2);
                        putvar(p1, val + p3);
                }
                break;
        case 5: // :=-
                if (!findvar(p2))
                        ok = 22;
                else {
                        val = getvar(p2);
                        putvar(p1, val - p3);
                }
                break;
        case 6: // =
                putvar(p1, p2);
                break;
        }
        return ok;
}
int Iloop::getloop(int *index) {
        if (loopEl_list.size() != 0) {
                LoopEl le = loopEl_list[loopEl_list.size() - 1];
                loopEl_list.pop_back();
                int count = le.get_count();
                count--;
                if (count < 1)
                        return *index;
                else {
                        int ind = le.get_index();
                        le.put_count(count);
                        loopEl_list.push_back(le);
                        return ind;
                }
        }
        else
                return *index; // указатель на текущий узел
}

int Iloop::checkloop() {
        if (loopEl_list.size() == 0)
                return 1;
        if (loopEl_list[loopEl_list.size() -1 ].get_count() < 1)
                return 0;
        else
                return 1;
}

int Iloop::putloop(int p1, int index) {
        if (findvar(p1)) {
                int val = getvar(p1);
                LoopEl le;
                le.put_count(val);
                le.put_index(index);
                loopEl_list.push_back(le);
                return 1;
        }
        else
                return 0;
}
```

```
void Iloop::putvar(int p1, int value) {
	int ok = 0;
	for (int i = 0; i < varEl_list.size(); i++) {
		if (varEl_list[i].get_index() == p1) {
			varEl_list[i].put_value(value);
			ok = 1;
			break;
		}

	}
	if (!ok) {
		VarEl ve;
		ve.put_index(p1);
		ve.put_value(value);
		varEl_list.push_back(ve);
	}
}

int Iloop::getvar(int p1) {
	int value = 0;
	for (int i = 0; i < varEl_list.size(); i++) {
		if (varEl_list[i].get_index() == p1) {
			value = varEl_list[i].get_value();
			break;
		}

	}
	return value;
}

int Iloop::findvar(int p1) {
	int value = 0;
	for (int i = 0; i < varEl_list.size(); i++) {
		if (varEl_list[i].get_index() == p1) {
			value = 1;
			break;
		}
	}
	return value;
```

7.5 Приложение 5 “Command.cpp”

```
#include "stdafx.h"
#include "Command.h"

Command::Command(int pns, string pcommandText, int pcommand, int perror, size_t pp1,
size_t pp2, size_t pp3)
{
	ns = pns;
	commandText = pcommandText;
	command = pcommand;
	error = perror;
	p1 = pp1;
	p2 = pp2;
	p3 = pp3;
}
void Command::set_command(int pns, string pcommandText, int pcommand, int perror, size_t
pp1, size_t pp2, size_t pp3) {
	ns = pns;
```

```
        commandText = pcommandText;
        command = pcommand;
        error = perror;
        p1 = pp1;
        p2 = pp2;
        p3 = pp3;
}
string Command::toString() {
        int ns = get_ns();
        string Ct = get_commandText();
        string te = get_textError(error);
    return to_string(ns)+ " " + Ct + " " + te + " ";
}
string Command::get_textError(int error) {
        string er = "";
        switch (error) {
        case 0:
                er = "\0";
                break;
        case 1:
                er = " er: 1 Garbage characters in the use of DO";
                break;
        case 2:
                er = "  er: 2 Garbage characters in the use of END";
                break;
        case 3:
                er = "  er: 3 Garbage characters in the use of LOOP-X";
                break;
        case 4:
                er = "  er: 4 Garbage characters in the index of LOOP x";
                break;
        case 5:
                er = "   er: 5 Garbage characters in the left part of +-";
                break;
        case 6:
                er = "   er: 5 Garbage characters in the left part of +-";
                break;
        case 7:
                er = "  er: 7 Garbage characters in the index of left part";
                break;
        case 8:
                er = "  er: 8 Plus or minus are not found";
                break;
        case 9:
                er = " er: 9 There is unrecognised command";
                break;
        case 10:
                er = " er: 10 Garbage characters in the use of x";
                break;
        case 11:
                er = " er: 11 Garbage characters in the index of x";
                break;
        case 12:
                er = "er: 12 Garbage characters in the right part of = ";
                break;
        case 13:
                er = "er: 13 There is unrecognised command";
                break;
        case 14:
                er = " er: 14 Count of DO greater than LOOP";
                break;
        case 15:
                er = " er: 15 Count of END greater than LOOP or DO";
                break;
        case 21:
```

```
            er = "er: 21 The x is not available";
            break;
        case 22:
            er = " er: 22 The x is not initialised";
            break;
        case 23:
            er = " er: 23 Garbage characters in the right part of +- \0";
            break;
        case 24:
            er = "  er: 24 Limit of index is exceeded \0";
            break;
        }
        return er;
```

7.6 Приложение 6 “Iloop.cpp”

```
#include "stdafx.h"
#include "Iloop.h"


void Iloop::printList(vector<Command> list, ofstream& F) {
        for (int i = 0; i < list.size(); i++) {
            cout << list[i].toString() << endl;
            F << list[i].toString() << endl;
        }
}

string Iloop::varToString() {
        /* stringstream s;
        for (int i = 0; i < varEl_list.size(); i++){
            s << varEl_list[i].toString();
        }
        string str = s.str();
        return str; */
        string s = "";
        for (int i = 0; i < varEl_list.size(); i++) {
            s =  s + varEl_list[i].toString() + " ";
        }
        return s;

}

string Iloop::loopToString() {
        string s = "";
        for (int i = 0; i < loopEl_list.size(); i++){
            s = s + loopEl_list[i].toString() + " ";
        }
        return s;
        }
```

7.7 Приложение 7 “LoopEl.cpp”

```
#include "stdafx.h"
#include "LoopEl.h"

string LoopEl::toString() {
    return to_string(count) + " from " + to_string(index_command) + "; ";
}
```

7.8 Приложение 8 “VarEl.cpp”

```
#include "stdafx.h"
#include "VarEl.h"

string VarEl::toString() {
        return "x" + to_string(index) + " = " + to_string(value) + "; ";
}
```

7.9 Приложение 9 “Command.h”

```
#pragma once
#include <string>
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
class Command
{
public:
	int get_ns() { return ns; }
	string get_commandText() { return commandText; }
	int get_error() { return error; }
	int get_command() { return command; }
	int get_p1() { return p1; }
	int get_p2() { return p2; }
	int get_p3() { return p3; }
	void put_error(int e) { error = e; }
	void put_ns(int e) { ns = e; }
	void put_command(int e) { command = e; }
	void put_p1(int e) { p1 = e; }
	void put_p2(int e) { p2 = e; }
	void put_p3(int e) { p3 = e; }

	void set_command(int pns, string pcommandText, int pcommand, int perror, size_t
pp1, size_t pp2, size_t pp3);
	Command(int pns, string pcommandText, int pcommand, int perror, size_t pp1, size_t
pp2, size_t pp3);

	string toString();
	string get_textError(int error);

private:
	int ns;			// номер строки
	string commandText; // строка с командой
	int command;		// номер команды
	int error;		// номер ошибки
	size_t p1;		// индекс переменной слева или переменной в LOOP
	size_t p2;		// индекс переменной справа или константа
	size_t p3;		// константа

};
```

7.10 Приложение 10 “Iloop.h”

```
#pragma once
#include <iostream>
#include <fstream>
#include <vector>
#include "LoopEl.h"
#include "VarEl.h"
#include "Command.h"
#include <algorithm>
#include <sstream>
using namespace std;
class Iloop
{
public:
	int loadListProg(ifstream& F);
	int loadListVar(ifstream& F);

	int checkSyntax(ofstream& F);
	Command checkpline(string text, int ns);
	Command checkvline(string text, int ns);
```

```
        void doProg(ofstream& F, int comlimit);
        int  doCommand(int c, int p1, int p2, int p3, int * index);
        int findvar(int p1);
        int getloop(int *index);
        int putloop(int p1, int index);
        void putvar(int p1, int value);
        int getvar(int p1);
        int checkloop();

        void printList(vector<Command> list, ofstream& F);
        string varToString();
    string loopToString();

private:
        vector<LoopEl> loopEl_list;
        vector<VarEl> varEl_list;
        vector<Command> prog_list;
        vector<Command> var_list;
};
```

7.11 Приложение 11 “LoopEl.h”

```
#pragma once
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
using namespace std;
class LoopEl
{

private:
        int count;
        int index_command;

public:
        string toString();
        int get_count() { return count; }
        int get_index() { return index_command; }
        void put_count(int c) { count = c; }
        void put_index(int index) { index_command = index; }
};
```

3.12 Приложение 12 “VarEl.h”

```
#pragma once
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
using namespace std;
class VarEl
{
private:
        int index;
        int value;
public:
        int get_value() { return value; }
        int get_index() { return index; }
        void put_value(int c) { value = c; }
        void put_index(int ind) { index = ind; }
        string toString();
};
```