МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт (факультет)

Кафедра

КУРСОВАЯ РАБОТА

по дисциплине Структурное программирование

на тему Конструирование моделей лексического и синтаксического анализа на языке C++

Выполнил студент группы

*группа*

направления подготовки (специальности)

*шифр, наименование*

*фамилия, имя, отчество*

Руководитель

*фамилия, имя, отчество*

*должность*

Дата представления работы

« » 20 г. Заключение о допуске к защите

Оценка

,

количество баллов

Подпись преподавателя

Череповец,

*год*

**Аннотация**

В данной курсовой работе рассмотрен процесс конструирования моделей лексического и синтаксического анализа. Проведено изучение и описание предметной области, для заданного задания. Помимо этого, детально рассматривается каждый этап выполнения.

Курсовая работа включает в себя четыре приложения: техническое задание, блок-схемы функций, код программы и руководство пользователя.

В техническом задании, в основном, говорится о непосредственной разработке программы. Руководство пользователя содержит подробные инструкции, необходимые пользователю.

Оглавление

1. Введение 4

2. Основная часть 5

2.1 Изучение и описание предметной области 5

2.2 Постановка задачи6

2.3 Выбор структур данных для решения поставленной задачи6

2.4 Логическое проектирование8

2.5 Физическое проектирование программного обеспечения8

2.6 Проектирование интерфейса9

2.7 Кодирование10

2.8 Тестирование10

3. Заключение12

4. Источники информации13

5. Приложение 1. Техническое задание15

6. Приложение 2. Блок-схемы основных функций19

7. Приложение 3. Текст программы22

8. Приложение 4. Руководство пользователя40

**Введение**

Данная работа посвящена конструированию моделей лексического и синтаксического анализа.

Целью курсовой работы является разработка программы в соответствии с заданием.

Для осуществления обозначенной цели необходимо:

* проанализировать задание на курсовую работу;
* определить требования к программному продукту;
* выбрать типы данных, необходимые для решения задачи;
* изучить и разработать алгоритмы, для решения задачи;
* разработать модульную структуру программы;
* написать код программы;
* выполнить тестирование.

Объект разработки – программа.

Среда разработки – Microsoft Visual Studio.

Методологической основой для написания программы послужило задание на курсовую работу.

**Основная часть**

**Изучение и описание предметной области**

По заданию на курсовую работу, необходимо:

1. Разработать грамматику для моделирования работы компилятора согласно своему варианту. Для синтаксического анализа – КС-грамматику, для лексического анализа – регулярную грамматику.
2. Построить лексический анализатор, который решает следующие задачи: выделяет из текста входной программы все лексемы, входящие в заданную языковую конструкцию Switch, удаляет лишние пробелы и комментарии из входной строки. Построение лексического анализатора должно быть выполнено на основе конечного автомата. По конечному автомату должна быть построена соответствующая ему регулярную грамматику.
3. Построить КС-грамматику, описывающую синтаксис инструкции языка программирования Switch.
4. Доказать возможность вывода заданной синтаксической конструкции с помощью грамматики разработанной в п.3. на основе заданного варианта распознавателя. Результатом работы синтаксического анализатора должно быть дерево вывода и соответствующее ему порождение рассматриваемой цепочки.
5. Путем эквивалентных преобразований должна быть получена приведенная грамматика.

Программа должна:

1. выделять из текста входной программы все лексемы, входящие в заданную языковую конструкцию оператора switch языка С++;
2. удалять лишние пробелы и комментарии из входной строки;
3. преобразовывать исходный текст программы в псевдокод;
4. преобразовывать исходный текст программы в дескрипторный код;
5. выводить сообщения об ошибках (тип ошибки и в какой строке).

Одной из причин для разработки является отсутствие решений аналогичных задач.

Для выполнения задания нам понадобится Microsoft Visual Studio.

**Постановка задачи**

Перечень исходных данных, их характеристики и способ представления:

1. Текст программы (ввод с клавиатуры, строка);

Набор выполняемых функций:

1. проверка числа на то, является ли оно беззаковым целым;
2. функция для перевода String^ в char\*;
3. анализ символа на принадлежность к латинскому алфавиту и знаку подчеркивания, к цифре или точке, к массиву разделителей;
4. функция для определения класса лексемы.

Перечень результатов, их характеристики и способ представления:

1. лексема (вывод в таблицу к которой лексема принадлежит, в виде строки);
2. псевдокод (вывод в listbox, в виде строки);
3. дескрипторный код (вывод в listbox, в виде строки).

Среда функционирования программного продукта:

1. Microsoft Windows Server 2003 или Microsoft Windows XP или Microsoft Windows 2000 с пакетом обновления 3 (SP3) или более поздние версии данных операционных систем;
2. Microsoft Visual Studio 2015 или более поздние версии.

**Выбор структур данных для решения поставленной задачи**

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight[1].

Таблица 1

Таблица идентификаторов

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип данных |
| Текст программы | AllTextProgram | строка |
| Дескрипторный код | DK | строка |
| Буфер для ДК | buffDK | строка |
| Псевдокод | PK | строка |
| Разделители | limiters | массив |
| Ключевые слова | reservedWords | строка |
| Результат | Result | функция |
| Анализ | Analys | функция |
| Проверка на беззнаковое число | isUint | функция |
| System String в Char | SystemStringToChar | функция |
| Массив для записи результата | mas0 | массив |
| Массив для записи результата | mas1 | массив |
| Массив для записи результата | mas2 | массив |
| Массив для записи результата | mas3 | массив |
| Счётчик | i | целое число |
| Счётчик для записи результата | sw0 | целое число |
| Счётчик для записи результата | sw1 | целое число |
| Счётчик для записи результата | sw2 | целое число |
| Счётчик для записи результата | sw3 | целое число |
| Тип лексемы | type | целое число |
| Логическая переменная | let | bool |
| Пустая строка | NULLstr | строка |
| 1 | 2 | 3 |

Продолжение табл. 1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Текст программы с listbox | s | строка |
| Преобразованный текст программы | x | строка |
| Строка для записи результата | mas0converted | строка |
| Строка для записи результата | mas1converted | строка |
| Строка для записи результата | mas2converted | строка |
| Строка для записи результата | mas3converted | строка |
| Буфер | str | строка |
| Строка для записи ДК | DKconverted | строка |
| Строка для записи ПК | PKconverted | строка |
| Переменная для записи состояния | state | целое число |
| Буфер | temp | строка |
| Тип лексемы | type | целое число |
| Проверка на цифру | isdigit | функция |
| Проверка на букву | isletter | функция |
| Проверка на разделитель | islimiter | функция |
| Проверка на ошибку | iserror | функция |
| Содержимое ошибочного кода | other | строка |
| Промежуточный счётчик | n | целое число |
| Промежуточный счётчик | m | целое число |

**Логическое проектирование**

Для написания программы, был разработан алгоритм, а по нему составлена упрощённая Блок-схема. Блок-схема программы представлена на Рис. 1.

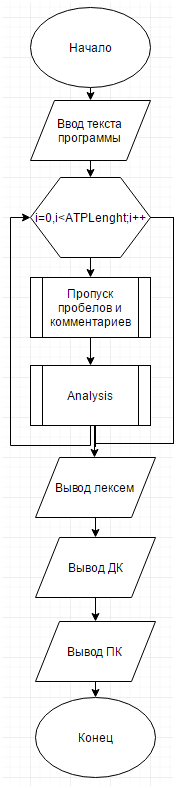


Рис. 1. Блок-схема

**Проектирование интерфейса**

Таблица 3

Таблица проектирования интерфейса

|  |  |  |
| --- | --- | --- |
| Элемент интерфейса | Реализация | Выполняемое действие |
| Поле для ввода текста программы | Поле listbox | Действий не выполняется |
| Кнопка ввод | Кнопка | Выполнение анализа текста введённого программы |
| Кнопка сброс | Кнопка | Возврат программы в изначальное состояние |
| Таблица ключевых слов | Таблица | Вывод результата |
| Таблица идентификаторов | Таблица | Вывод результата |
| Таблица числовых переменных | Таблица | Вывод результата |
| Таблица знаков пунктуации | Таблица | Вывод результата |
| Поле для вывода дескрипторного кода | Поле listbox | Вывод результата |
| Поле для вывода псевдокода | Поле listbox | Вывод результата |

**Кодирование**

Реализация разработанных алгоритмов и составленный по ним текст программы (с комментариями) в *Приложении 3*.

**Тестирование**

На этапе тестирования разработаны тестовые данные и оформлены в виде таблицы.

Таблица 4

Таблица тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Исходные данные | Тестируемый модуль или подпрограммы | Ожидаемый результат |
| 1 | int x=5; | MyForm | Выделение лексем |
| 2 | int xQx3=5.5; | MyForm | Выделение лексем |
| 3 | switch (i) { break; }; | MyForm | Выделение лексем |
| 4 | int x=5; //sadsad | MyForm | Пропуск |
| 5 | /\*int x=5;\*/ | MyForm | Пропуск |

**Проведение тестирования по таблице тестовых данных**

Таблица 5

Результаты выполнения тестирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Дата и время | Тестируемый модуль или подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 1 | 9.01.2017 | MyForm | Кулагин А. | Ввод текста программы для проверки на определение лексем: int x=5; | Успех |
| 2 | 9.01.2017 | MyForm | Басистюк А.. | Ввод текста на проверку определения введённых переменных и констант: int xQx3=5.5; | Успех |
| 3 | 9.01.2017 | MyForm | Осипов А. | Проверка на определение конструкций оператора Switch:  switch (i) { break; }; | Успех |
| 4 | 9.01.2017 | MyForm | Борисов С. | Проверка на пропуск пробелов и комментариев:  int x=5; //sadsad | Успех |
| 5 | 9.01.2017 | MyForm | Стаськевич Л. | Проверка на пропуск комментария:  /\*int x=5;\*/ | Успех |

**Заключение**

В ходе выполнения курсовой работы была разработана грамматика для моделирования работы компилятора. Для синтаксического анализа – КС-грамматика, для лексического анализа – регулярную грамматика.

При написании курсового проекта была изучена специальная литература, включающая в себя статьи, учебники, руководства, описаны теоретические аспекты и раскрыты ключевые понятия исследования.

Во время написания курсовой работы был решён ряд поставленных задач:

* проанализировали задание на курсовую работу;
* определили требования к программному продукту;
* выбрали типы данных, необходимые для решения задачи;
* изучили и разработали алгоритмы, для решения задачи;
* разработали модульную структуру программы;
* написали код программы;
* выполнили тестирование.

В том числе выполнено техническое задание.

Результатом курсовой работы стала программа, написанная на C++.

**Источники информации**

1. Майкрософт [Электронный ресурс] //Microsoft. URL: <https://www.microsoft.com/> (дата обращения: 25.12.2016).

2. Майкрософт [Электронный ресурс] //Microsoft. URL: <https://msdn.microsoft.com/ru-ru/library/dn762121.aspx> (дата обращения: 25.12.2016).

*ПРИЛОЖЕНИЕ 1*

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

наименование института (факультета)

наименование кафедры

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой ,

д.т.н., профессор Ершов Е.В.

« » 20 г.

Конструирование моделей лексического и синтаксического анализа на языке C++

Техническое задание на курсовую работу

Листов

Руководитель

Ф.И.О. преподавателя

Исполнитель

студент

группа

Фамилия, имя, отчество

год

**Введение**

В процессе написания курсовой работы необходимо написать программу на языке программирования C++, моделирующую работу лексического и синтаксического анализатора. Средой для разработки служит Microsoft Visual Studio - набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию[2].

**Основания для разработки**

Задание на Курсовую работу по Теории автоматов и формальных языков. Череповецкий Государственный Университет 2016 год.

**Назначение разработки**

Программа должна моделировать работу компилятора для оператора switch языка С++.

**Требования к программе**

*Требования к функциональным характеристикам:*

1. Программа должна выделять из текста входной программы все лексемы, входящие в заданную языковую конструкцию оператора switch языка С++;
2. Программа должна удалять лишние пробелы и комментарии из входной строки;
3. Программа должна преобразовывать исходный текст программы в псевдокод;
4. Программа должна преобразовывать исходный текст программы в дескрипторный код;
5. Программа должна выводить сообщения об ошибках (тип ошибки и в какой строке).

*Требования к надежности:*

* программа должна выполнять предписанные функциональные характеристики без сбоев;
* обеспечение контроля входной и выходной информации;
* защита при неверных действиях пользователя;
* контроль соответствия типов данных.

*Условия эксплуатации:*

* компьютер, поддерживающий операционную систему Windows XP;
* наличие на компьютере операционной системы Windows XP или более поздней версии;
* знание основ работы в операционной системе Windows.

*Требования к составу и параметрам технических средств:*

* для запуска программы требуется Windows XP или более поздняя версия и Super VGA видеоадаптер;
* процессор 233 МГц или лучше;
* как минимум 64 МБ Мб ОЗУ;
* не менее 1,5 ГБ свободного дискового пространства;
* для установки Windows требуется устройство для чтения компакт-дисков (или же поддержка других устройств, таких как флэш-накопителей);
* необходим монитор Super VGA с разрешением 800x600 или более высоким, отображающий 256 и более цветов;
* необходимы клавиатура и мышь.

*Требования к информационной и программной совместимости:*

* язык программирования (C++);
* среда для разработки программы (Visual Studio C++)
* операционная система (Windows XP);
* уровень защиты (без защиты).

*Требования к маркировке и упаковке:*

Распространение на электронных носителях или через средства коммуникации (для некоммерческого использования).

*Требования к транспортированию и хранению:*

Без специальных требований.

**Требования к программной документации**

*Наличие различной документации:*

* руководства пользователя;
* текст программы;
* расчетно-пояснительная записка с приложениями.

**Технико-экономические показатели**

Преимущества разработки по сравнению с существующими отечественными и зарубежными аналогами – программа должна быть простой в использовании и выполняться последовательно с описанием каждого этапа.

**Стадии и этапы разработки**

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки ПО | Сроки разработки | Результат выполнения | Отметка  о выполнении |
| Получение задания | 9.10.2016 | Выполнен успешно | Выполнено |
| Анализ требований | 10.10.16-13.10.16 | Выполнен успешно | Выполнено |
| Реализация | 14.10.16-19.10.16 | Выполнен успешно | Выполнено |
| Тестирование | 20.10.16-24.10.16 | Выполнен успешно | Выполнено |
| Внедрение и поддержка | 25.10.16-29.10.16 | Выполнен успешно | Выполнено |

**Порядок контроля и приемки**

Таблица П1.2

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Техническое задание | 20.12.16 | Успех | Выполнен |
| Написание программы | 24.12.16 | Успех | Выполнен |
| Расчётно-пояснительная записка | 25.12.16 | Успех | Выполнен |
| Руководство пользователя | 25.12.16 | Успех | Выполнен |

*ПРИЛОЖЕНИЕ 2*

**Блок-схемы основных функций**

Функции isdigit, isletter, islimiter, iserror – аналогичны, за исключением условия. В функции isdigit идёт проверка на принадлженость к цифре. Эта функция изображена на Рис. П2.1. В функции isletter идёт проверка на принадлженость к букве и знаку подчёркивания. В функции islimiter проверка на принадлженость к разделителю. В функции iserror проверка на запрещённый символ.

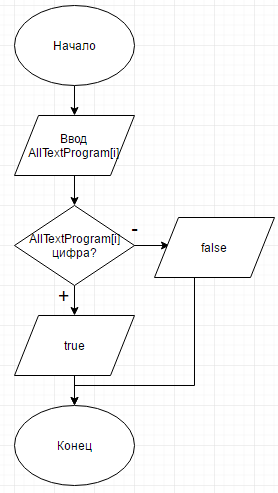


Рис. П2.1. Блок-схема функции isdigit

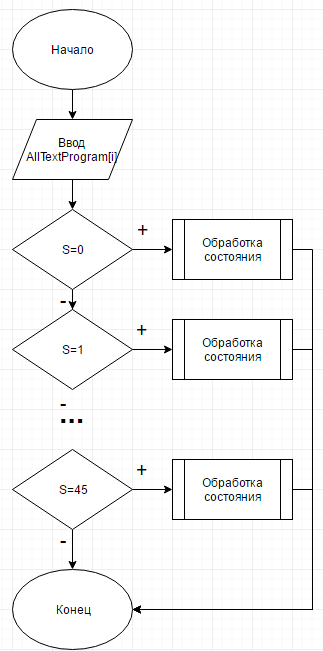


Рис. П2.2. Блок-схема функции Analys

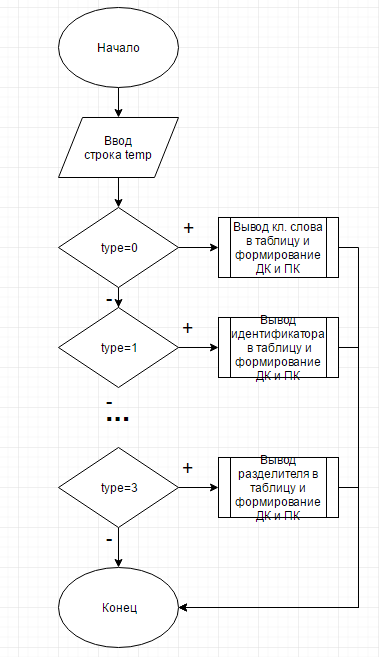


Рис. П2.3. Блок-схема функции Result

*ПРИЛОЖЕНИЕ 3*

**Текст программы**

//функции

static void Analys(char nextChar); //анализ

static void Result(std::string temp); //результат

bool isdigit(char l); //проверка на цифру

bool isletter(char l); //проверка на букву

bool islimiter(char l); //проверка на разделитель

bool iserror(char l); //проверка на ошибку

char\* SystemStringToChar(System::String^ string);

//массивы для записи результата

std::string mas0[100];

std::string mas1[100];

std::string mas2[100];

std::string mas3[100];

std::string DK; //строка для записи дескрипторного кода

std::string buffDK; //буфер для записи дескрипторного кода

std::string PK = ""; //строка для псевдокода

std::string other="Строка с ошибкой: "; //строка для неопознанных лексем

int i = 0; //счётчик

int sw0 = 0, sw1 = 0, sw2 = 0, sw3 = 0; // переменные счётчики для записи результата

int type=4; // переменная для определения типа лексемы

std::string temp=""; //буфер

std::string NULLstr; //пустая строка

std::string AllTextProgram; //строка для записи текста программы

int n, m = 0; //счётчики

int state = 0; //определение начального состояния

bool isdigit(char l) {

bool let = false;

std::string a = "0123456789";

for (int k = 0; k <= a.length(); k++)

{

if (a[k] == l)

let = true;

}

if (let == true)

return true;

else

return false;

}

bool isletter(char l) {

bool let = false;

std::string a = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz\_";

for (int k = 0; k <= a.length(); k++)

{

if (a[k] == l)

let = true;

}

if (let == true)

return true;

else

return false;

}

bool islimiter(char l) {

bool let = false;

std::string a = ",.()[]:;+-\*/<>@={} ";

for (int k = 0; k <= a.length(); k++)

{

if (a[k] == l)

let = true;

}

if (let == true)

return true;

else

return false;

}

bool iserror(char l) {

bool let = false;

std::string a = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЪЭЮЯабвгдеёжзийклмнопрстуфхцчшщьыъэюя|~`@№^?&!\_";

for (int k = 0; k <= a.length(); k++)

{

if (a[k] == l)

let = true;

}

if (let == true)

return true;

else

return false;

}

static void Analys(char nextChar) {

switch (state) {

case 0: {

if (AllTextProgram[i] == 'i') { state = 6; type = 0; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == 'c') { state = 10; type = 0; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == 's') { state = 19; type = 0; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == 'b' ) { state = 25; type = 0; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == 'd') { state = 30; type = 0; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == 'e') { state = 38; type = 0; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == 't') { state = 42; type = 0; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 2; type = 2; temp += AllTextProgram[i]; }

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i];}

else { state = 0; m = i; }

break; }

case 1: {

if (isletter(AllTextProgram[i])) { state = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' '||AllTextProgram[i]=='\0'||AllTextProgram[i]=='\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i];}

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 2: {

if (isdigit(AllTextProgram[i])) { state = 2; type = 2; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '.') { state = 3; type = 2; temp += AllTextProgram[i]; }

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i];}

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 3: {

if (isdigit(AllTextProgram[i])) { state = 4; temp += AllTextProgram[i];}

else { state = 5; }

break; }

case 4: {

if (isdigit(AllTextProgram[i])) { state = 4; temp += AllTextProgram[i];}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (iserror(AllTextProgram[i])) { state =5; temp += AllTextProgram[i];}

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 5: {

if (isletter(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i];}

else if (isdigit(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; other += temp; other += " !!! "; temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; other += temp; other += " !!! "; temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else { state = 5; temp += AllTextProgram[i];}

break; }

case 6: {if (AllTextProgram[i] == 'n') { state = 7; type = 0; temp += AllTextProgram[i];}

else if (AllTextProgram[i] == 'f') { state = 37; type = 0; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 7: {if (AllTextProgram[i] == 't') { state = 8; temp += AllTextProgram[i];}

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 8: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 10: {if (AllTextProgram[i] == 'a') { state = 11; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 11: {if (AllTextProgram[i] == 's') { state = 12; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 12: {if (AllTextProgram[i] == 'e') { state = 13; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 13: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 19: {if (AllTextProgram[i] == 'w') { state = 20; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 20: {if (AllTextProgram[i] == 'i') { state = 21; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 21: {if (AllTextProgram[i] == 't') { state = 22; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 22: {if (AllTextProgram[i] == 'c') { state = 23; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 23: {if (AllTextProgram[i] == 'h') { state = 24; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 24: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 25: {if (AllTextProgram[i] == 'r') { state = 26; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 26: {if (AllTextProgram[i] == 'e') { state = 27; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 27: {if (AllTextProgram[i] == 'a') { state = 28; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 28: {if (AllTextProgram[i] == 'k') { state = 29; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 29: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 30: {if (AllTextProgram[i] == 'e') { state = 31; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 31: {if (AllTextProgram[i] == 'f') { state = 32; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 32: {if (AllTextProgram[i] == 'a') { state = 33; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 33: {if (AllTextProgram[i] == 'u') { state = 34; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 34: {if (AllTextProgram[i] == 'l') { state = 35; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 35: {if (AllTextProgram[i] == 't') { state = 36; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 36: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 37: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 38: {if (AllTextProgram[i] == 'l') { state = 39; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 39: {if (AllTextProgram[i] == 's') { state = 40; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 40: {if (AllTextProgram[i] == 'e') { state = 41; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 41: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 42: {if (AllTextProgram[i] == 'h') { state = 43; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 43: {if (AllTextProgram[i] == 'e') { state = 44; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 44: {if (AllTextProgram[i] == 'n') { state = 45; temp += AllTextProgram[i]; }

else if (isletter(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; type = 1; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; type = 1; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

case 45: {if (isletter(AllTextProgram[i])) { state = 1; type=1; temp += AllTextProgram[i]; }

else if (isdigit(AllTextProgram[i])) { state = 1; type = 1; temp += AllTextProgram[i]; }

else if (AllTextProgram[i] == '=' && AllTextProgram[i + 1] == '=') {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; temp += AllTextProgram[i + 1]; type = 3; Result(temp); temp = NULLstr; i++;

}

else if (islimiter(AllTextProgram[i])) {

state = 0; Result(temp); temp = NULLstr;

temp += AllTextProgram[i]; type = 3; Result(temp); temp = NULLstr; type = 4;

}

else if (AllTextProgram[i] == ' ' || AllTextProgram[i] == '\0' || AllTextProgram[i] == '\n') {

state = 0; Result(temp); temp = NULLstr;

}

else if (iserror(AllTextProgram[i])) { state = 5; temp += AllTextProgram[i]; }

else {

state = 0; other += AllTextProgram[i];

}

break; }

}

return;

}

static void Result(std::string temp) {

switch (type)

{

case 0:

for (int j = 0; j < 100; j++)

{

if (temp == mas0[j])

{

PK += temp;

buffDK = std::to\_string(j + 1);

DK += "(10,"; DK += buffDK; DK += ")";

return;

}

}

PK += temp;

buffDK = std::to\_string(sw0 + 1);

DK += "(10,"; DK += buffDK; DK += ")";

mas0[sw0] = temp;

sw0++;

return;

break;

case 1:

for (int j = 0; j < 100; j++)

{

if (temp == mas1[j])

{

PK += "id";

buffDK = std::to\_string(j + 1);

DK += "(20,"; DK += buffDK; DK += ")";

return;

}

}

PK += "id";

buffDK = std::to\_string(sw1 + 1);

DK += "(20,"; DK += buffDK; DK += ")";

mas1[sw1] = temp;

sw1++;

return;

break;

case 2:

for (int j = 0; j < 100; j++)

{

if (temp == mas2[j])

{

PK += "const";

buffDK = std::to\_string(j + 1);

DK += "(30,"; DK += buffDK; DK += ")";

return;

}

}

PK += "const";

buffDK = std::to\_string(sw2 + 1);

DK += "(30,"; DK += buffDK; DK += ")";

mas2[sw2] = temp;

sw2++;

return;

break;

case 3:

for (int j = 0; j < 100; j++)

{

if (temp == mas3[j])

{

PK += temp;

buffDK = std::to\_string(j + 1);

DK += "(40,"; DK += buffDK; DK += ")";

return;

}

}

PK += temp;

buffDK = std::to\_string(sw3 + 1);

DK += "(40,"; DK += buffDK; DK += ")";

mas3[sw3] = temp;

sw3++;

return;

break;

}

}

char\* SystemStringToChar(System::String^ string) {

return (char\*)(void\*)System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(string);

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

//Создание таблицы ключевых слов

dataGridView1->ColumnCount = 3;

dataGridView1->RowCount = 1;

//

//заголовки столбцов

dataGridView1->Columns[0]->HeaderCell->Value = "10";

dataGridView1->Columns[1]->HeaderCell->Value = "Кл. сл.";

dataGridView1->Columns[2]->HeaderCell->Value = "п/к";

//

//выравнивание

dataGridView1->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

dataGridView1->AutoResizeColumns();

//

//

//Создание таблицы индетификаторов

dataGridView2->ColumnCount = 3;

dataGridView2->RowCount = 1;

//

//заголовки столбцов

dataGridView2->Columns[0]->HeaderCell->Value = "20";

dataGridView2->Columns[1]->HeaderCell->Value = "Идент.";

dataGridView2->Columns[2]->HeaderCell->Value = "п/к";

//

//выравнивание

dataGridView2->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

dataGridView2->AutoResizeColumns();

//

//

//Создание таблицы числовых переменных

dataGridView3->ColumnCount = 3;

dataGridView3->RowCount = 1;

//

//заголовки столбцов

dataGridView3->Columns[0]->HeaderCell->Value = "30";

dataGridView3->Columns[1]->HeaderCell->Value = "Числ. перем.";

dataGridView3->Columns[2]->HeaderCell->Value = "п/к";

//

//выравнивание

dataGridView3->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

dataGridView3->AutoResizeColumns();

//

//

//Создание таблицы знаков

dataGridView4->ColumnCount = 3;

dataGridView4->RowCount = 1;

//

//заголовки столбцов

dataGridView4->Columns[0]->HeaderCell->Value = "40";

dataGridView4->Columns[1]->HeaderCell->Value = "Зн. оп. пункт.";

dataGridView4->Columns[2]->HeaderCell->Value = "п/к";

//

//выравнивание

dataGridView4->AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders);

dataGridView4->AutoResizeColumns();

//

//заполнение label

label1->Text = "Исходный текст программы";

label2->Text = "Дескрипторный код";

label3->Text = "Псевдокод";

label4->Text = "";

label5->Text = "";

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

String ^s="";

//s = textBox1->Text;

s = richTextBox3->Text;

std::string str;

char\* x = SystemStringToChar(s);

AllTextProgram = x;

//

//Тело программы

//

for (int j = 0; j < 100; j++)

{

mas0[j] = ""; mas1[j] = ""; mas2[j] = ""; mas3[j] = "";

};

for (i = 0; i < AllTextProgram.length(); i++)

{

char c = AllTextProgram[i];

if (AllTextProgram[i] == '\'')

{

temp += '\'';

i++;

while (AllTextProgram[i] != '\'')

{

temp += AllTextProgram[i];

i++;

}

temp += '\'';

type = 2;

Result(temp);

temp = NULLstr;

}

if (AllTextProgram[i] == '/'&&AllTextProgram[i + 1] == '\*')

{

int chet = 1;

while (chet != 0)

{

i++;

if (AllTextProgram[i] == '/'&&AllTextProgram[i + 1] == '\*')

chet++;

if (AllTextProgram[i] == '\*'&&AllTextProgram[i + 1] == '/')

chet--;

} i = i + 2;

}

if (AllTextProgram[i] == ' '&&AllTextProgram[i + 1] == ' ')

{

Analys(AllTextProgram[i]);

int chet = 1;

while (chet != 0)

{

i++;

if (AllTextProgram[i] != ' ')

chet--;

}

}

//

if (AllTextProgram[i] == '/'&&AllTextProgram[i + 1] == '/')

{

int chet = 1;

while (chet != 0)

{

i++;

if (AllTextProgram[i] == '\n'||AllTextProgram[i]=='\0')

chet--;

}

}

Analys(AllTextProgram[i]);//анализ кода

}

for (int j = 0; j < 100; j++)

{

String^ mas0converted = gcnew String(mas0[j].c\_str());

dataGridView1->Rows->Add(j+1, mas0converted, mas0converted);

String^ mas1converted = gcnew String(mas1[j].c\_str());

dataGridView2->Rows->Add(j + 1, mas1converted, "id");

String^ mas2converted = gcnew String(mas2[j].c\_str());

dataGridView3->Rows->Add(j + 1, mas2converted, "const");

String^ mas3converted = gcnew String(mas3[j].c\_str());

dataGridView4->Rows->Add(j + 1, mas3converted, mas3converted);

}

String^ DKconverted = gcnew String(DK.c\_str());

String^ PKconverted = gcnew String(PK.c\_str());

richTextBox1->Text = DKconverted;

richTextBox2->Text = PKconverted;

String^ label44 = gcnew String(other.c\_str());

label4->Text = label44;

}

*ПРИЛОЖЕНИЕ 4*

**Руководство пользователя**

**Общие сведения о программе**

Программа на C++, моделирует работу компилятора для оператора Switch.

**Описание установки**

Для использования программы потребуются:

1. Операционная система Windows Server 2008 SP2 (x86 и x64) или более поздняя версия;
2. Microsoft .NET Framework 3.0 или более поздняя версия.

**Инструкции по работе**

После запуска вы увидите интерфейс программы. Он изображён на Рис. П4.1.

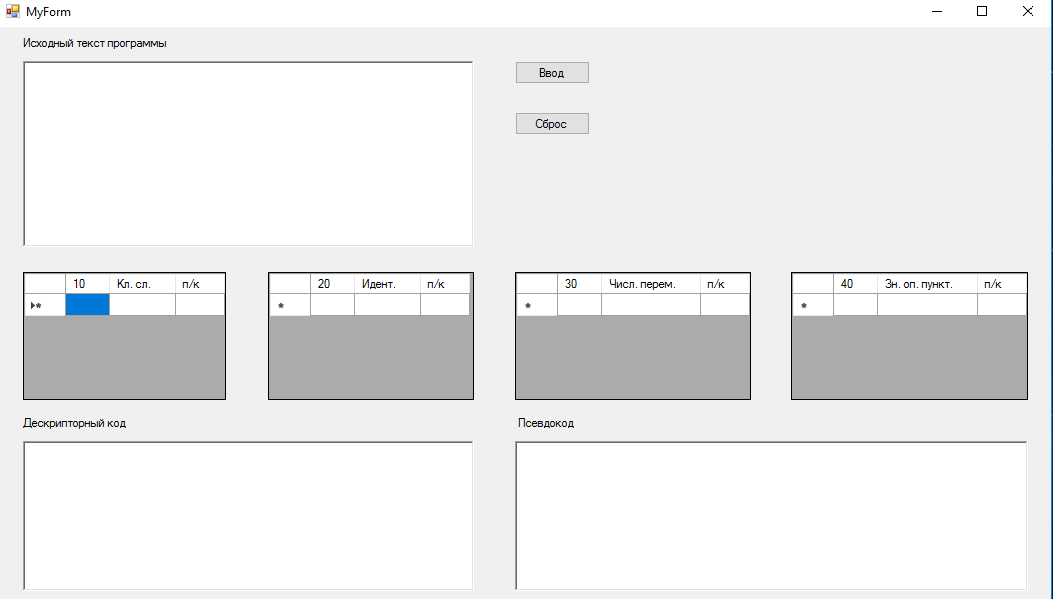


Рис. П4.1. Интерфейс программы

Ввод текста программы производится в соответствующее окно, находящееся в левом верхней части интерфейса. После нажатия на кнопку “Ввод” программа выделяет лексемы и выводит дескрипторный код и псевдокод в соответствующие поля. Результат представлен на Рис. П4.2.

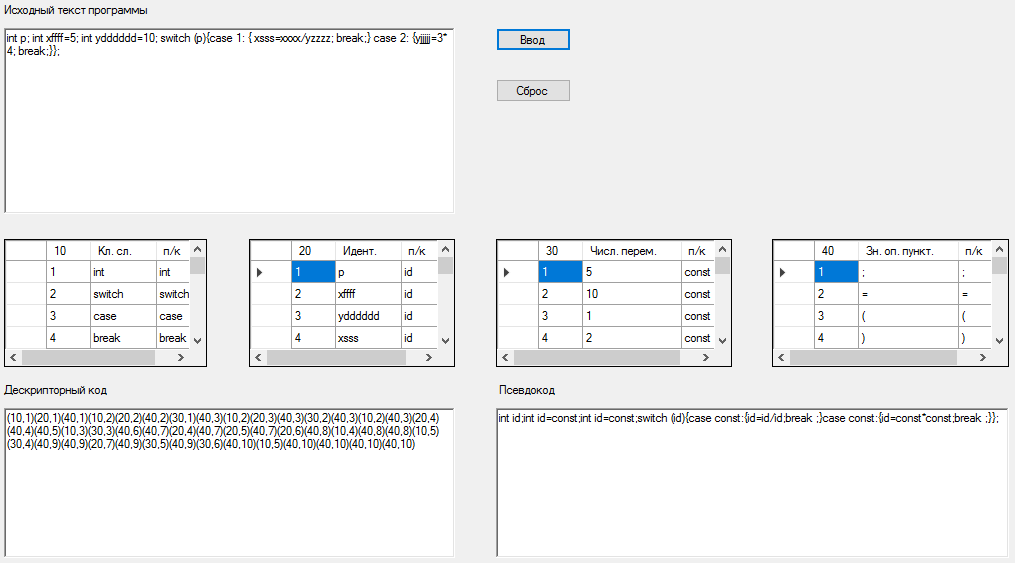


Рис. П4.2. Результат выполнения программы

Возврат программы в изначальное состояние производится при помощи нажатия на кнопку “Сброс”.