[1. Введение 2](#_Toc91085739)

[1.1.Проблема, цель и задачи 4](#_Toc91085740)

[2.Техническая составляющая задачи. 5](#_Toc91085741)

[2.1. Средства реализации 5](#_Toc91085742)

[2.3. Минимальные требования к системе команд 6](#_Toc91085743)

[3. Реализация 6](#_Toc91085744)

[3.1. Список команд 6](#_Toc91085745)

[3.2. Логическая схема 9](#_Toc91085746)

[3.3. Ограничения 9](#_Toc91085747)

[4. Тестирование 10](#_Toc91085748)

[5. Заключение 11](#_Toc91085749)

[Приложение 12](#_Toc91085750)

[Список литературы: 23](#_Toc91085751)

**1. Введение**

Прежде чем начать говорить о написании собственного языка программирования (ЯП) стоит сказать о том, что такое ЯП. Язык программирования – формальный язык, с помощью которого происходит обмен данными (“общение”) между программистом и компьютером [1]. Как и в любом другом языке ЯП имеет свой синтаксис – то, как должны записываться команды на ЯП, и семантику – правила в соответствии с которыми осуществляется выполнение того или иного набора инструкций [2]. В эпоху развития интернета количество языков программирования растёт с каждым годом. Некоторые из них становятся популярными и их используют программисты по всему миру, например: Java, Python, C++, C#, а некоторые становятся локальными и почти никому неизвестными [3].

В каждом ЯП существуют две обязательные части: лексический и синтаксический анализатор. Лексический анализатор – разбор входного потока символов на токены [4]. Синтаксический анализатор или парсер – часть программы, анализирующая данные полученные из лексического анализатора и преобразующая текст, записанный на ЯП, во внутреннее представление, которое будет удобно считывать транслятору [5]. Исходный код парсера может быть сгенерирован например, через Yacc, GNU Bison или же написан программистом.

А теперь давайте выясним, зачем же вообще создавать новые ЯП. Как правило, каждый ЯП призван решать какую-то конкретную задачу, которую не удаётся реализовать на других языках, например: Python хорошо подходит для работы с большим объёмом данных, ведь справляется с этой задачей быстрее других [6].

Именно по этой причине создание новых ЯП не прекратится никогда, ведь области, в которых нужен будет специальный ЯП огромное количество, к тому же появляются и новые области.

В рамках курсовой работы я попытался создать язык программирования, который мог бы использоваться в системе умного дома [7]. В создании собственного ЯП важной частью является транслятор. Транслятор выполняет функцию перевода программы с созданного ЯП на целевой язык. В моём случае транслятор должен переводить команды, написанные на созданном мною ЯП, в текстовый формат понятный человеку для упрощения работы с умным домом [8].

Умный дом – технологическая новинка, которая только начинает своё распространение по миру [9]. Умный дом – система автоматизации для управления поведением устройств, находящихся в доме. Сценарии – это ключевая особенность умных домов, ведь они позволяют не выполнять человеку рутинную работу по дому, например: выключить все устройства перед сном, включая свет и газ, включить обогреватель, если температура в доме упала до какого-то заранее заданного значения и т.д [10].

Таким образом, ЯП должен позволять выполнять различные действия с устройствами, а также иметь сценарии для работы с этими устройствами. Также стоит дополнить, что коммуникация с умным домом должна осуществляться понятным человеку образом, т.е чтобы даже не программист мог также его понять.

Для начала работы необходимо было выбрать список, описываемых устройств, чтобы заранее можно было понять, какие сценарии на их основе можно создать. После того, как я определился с типом устройств и тем, какие функции они будут иметь, я перешёл к сценариям. Для себя я определил, что параметры сценариев должны быть изменяемыми, чтобы каждый пользователь мог отредактировать их под себя. Ещё стоит отметить, что коммуникация с умным домом происходит на английском языке, что является, конечно, ограничением по применению данного языка в России и других странах, где английский язык изучается не во всех школах.

Также ограничениями по применению могут быть законодательства стран, в которых используются умные дома, ведь технология ещё является новой и не всем понятно, как регулировать её с точки зрения закона, например: кто будет виноват в случае пожара в умном доме? Человек, который забыл выключить газ или система, которая по какой-то причине не выключила его во время.

## **1.1. Проблема, цель и задачи**

Цель работы: написать собственный язык программирования для взаимодействия с системой умного дома посредством текстовых файлов.

Задачи:

1. Изучить и выбрать способ реализации собственного языка программирования.
2. Выявить плюсы и минусы найденного способа.
3. Придумать и реализовать грамматику языка и набор инструкций.
4. Провести тестирование взаимодействия с системой умного дома.

# **2.Техническая составляющая задачи.**

В качестве целевого языка программирования (ЯП), на котором будет осуществлять работу наш ЯП, был выбран С, так как одним из главных его качеств является скорость работы. Инструментами, с помощью которого и будет создаваться наш ЯП являются Yacc и Lex, потому что они удобны в использовании, ведь изначально и создавались для написания собственного ЯП.

## **2.1. Средства реализации**

Что такое Yacc? Yacc(yet another compiler-compiler) - это генератор синтаксического анализатора LALR. Yacc изначально был разработан для дополнения Lex. Генератор синтаксического анализатора - это программа, которая принимает в качестве входных данных спецификацию синтаксиса и выдает на выходе процедуру распознавания этого языка. Исторически их еще называют компиляторами-компиляторами.

Что такое Lex? Lex - это программа, которая генерирует лексический анализатор. Он используется с генератором синтаксического анализатора Yacc. Лексический анализатор - это программа, которая преобразует входной поток в последовательность токенов.

Токен – будущая структурная единица языка, который мы хотим написать. Например, в Java это: if, else, for и т.д, в токены также входят константы и переменные.

Lex считывает входной поток и выдает исходный код в качестве выходного посредством реализации лексического анализатора в программе на языке С. Функция Lex заключается в следующем:

1. Сначала лексический анализатор создает программу .lex на языке Lex. Затем компилятор Lex запускает программу .lex и создает программу на языке C “lex.yy.c”.
2. Наконец, компилятор C запускает программу “lex.yy.c” и создает объектную программу “a.out”.
3. “a.out” - это лексический анализатор, который преобразует входной поток в последовательность токенов.

Lex и Yacc вместе работают следующим образом: изначально в Lex записывается какой-то набор правил, на основании которых мы будем анализировать код нашего языка, иначе говоря, это лексика языка. В Yacc же необходимо записать набор инструкций, которые будут выполняться, если встретится какая-то определённая последовательность токенов. Таким образом, процесс компиляции выглядит следующим образом:

1. Lex анализирует лексику языка.
2. Далее передаёт поток токенов в Yacc.
3. В Yacc если встречается поток токенов, соответствующий какой-либо инструкции, выполняется инструкция.
4. Выводится результат работы программы.

## **2.3. Минимальные требования к системе команд**

Язык управления сценариями умного дома. В умном доме находится не менее 5 типов устройств (датчиков и исполнительных устройств), каждое из которых может выполнять несколько действий. Транслятор выводит в выходной файл состояние всех устройств в каждый момент времени.

**3. Реализация**

Проанализировав особенности работы с Yacc, Lex и особенности компиляции на Linux, я написал список команд, с помощью которых может происходить управление устройствами и датчиками умного дома, и несколько сценариев, которые будут срабатывать при различных состояниях устройств в доме.

**3.1. Список команд**

Команды для работы с устройствами:

1. Обогреватель

* heater on - вкл обогревателя
* heater off - выкл обогревателя
* heater target temperature N, где N - число, устанавливается температура обогревателя равная N

1. Робот-пылесос:

* cleaner on - вкл робот-пылесос
* cleaner off - выкл робот-пылесос
* cleaner silent - выбрать тихий режим работы для пылесоса
* cleaner wet - выбрать влажный режим уборки для пылесоса
* cleaner normal - выбрать обычный режим работы для пылесоса

Команды для работы с датчиками:

1. Входная дверь:

* door:unlock - дверь разблокирована
* door:lock - дверь заблокирована

1. Термометр :

* thermometer: N°C, где N число - измеряет температуру в доме

1. Окна:

* windows:open - окна открыты
* windows:close - окна закрыты

1. Свет:

* light:N% , где N число, измеряется количество солнечного света на улице

1. Шум:

* noise:N%, где N число, измеряется уровень шума на улице

1. Движения в доме:

* motion:true - движения обнаружены
* motion:false - движения не обнаружены

Сценарии:

1. Сон:

* Если движения не обнаружены, уровень освещённости < 10%, уровень шума < 10%, то запускается сценарий работы дома: сон, тогда изменяется состояние устройств: если робот-пылесос включен не в тихом режиме работы, то он принудительно выключается

1. Отопление:

* Если температура в доме < 20°C, то запускается сценарий “отопление”, тогда изменяется состояние устройств: окна закрываются, обогреватель включается с температурой 30°C

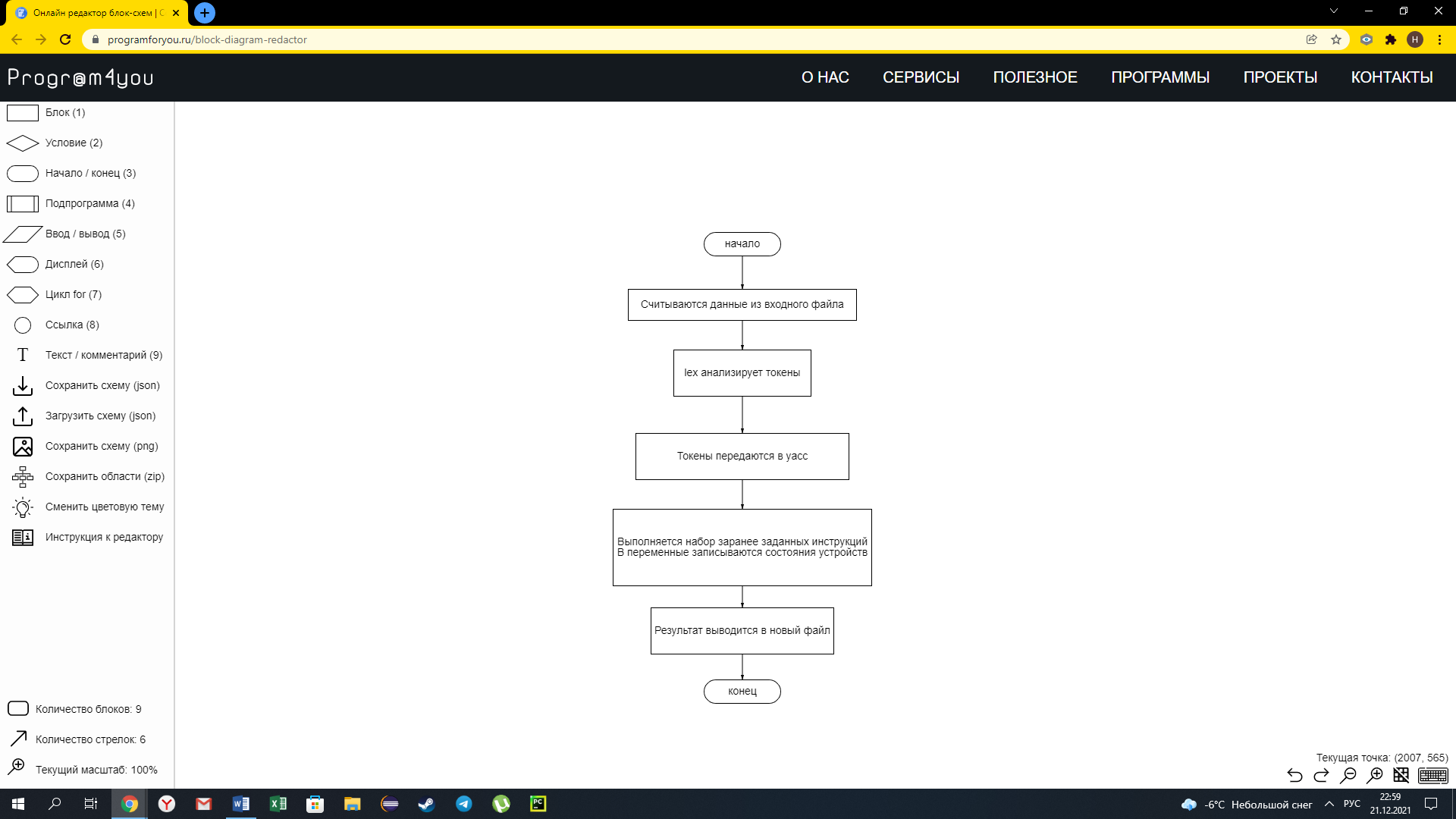
1. Дом пуст:

* Если движения не обнаружены и входная дверь заблокирована, то срабатывает сценарий работы “дом пуст”

Редактирование сценариев:

* edit action thermometer:N, где N число, изменяет параметр температуры у сценария
* edit action light:N, где N число, изменяет параметр освещения у сценария
* edit action noise:N, где N число, изменяет параметр шума у сценария

## **3.2. Логическая схема**



## **3.3. Ограничения**

К ограничениям связанным с применением языка относятся:

1. В языке нельзя создавать собственные новые сценарии.
2. Нет возможности добавления новых типов устройств.
3. Команды записываются на английском языке.

# **4. Тестирование**

Результаты тестирования программы:

|  |  |
| --- | --- |
| Исходные данные | Результат работы |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# **5. Заключение**

Резюмируя, можно сказать, что в рамках курсовой работы мне удалось реализовать собственный язык программирования для системы умного дома. Как и было сказано выше, инструменты Yacc и Lex упрощают создание собственного языка программирования, так как с их помощью легче анализировать входные данные. В процессе разработки я столкнулся с несколькими проблемами. Первая из них заключалась в том, что встроенный редактор кода на ОС Linux не очень удобный в использовании, так как иногда создаются ничем необоснованные ошибки самой программой. А вторая проблема – отсутствие какой-либо информации на русском языке о том, что такое Yacc и Lex, поэтому пользоваться приходилось исключительно англоязычными сайтами.

# **Приложение**

Исходный код синтаксического анализатора Yacc:

%{

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

extern FILE \*yyin;

FILE \*outputFile;

struct system {

int light;

int noise;

char door [50];

char windows [50];

char motion [50];

long int ttime;

int thermometer;

};

struct system home;

struct system2{

char \*cleaner;

char \*heater;

int theater;

};

struct system2 devices;

%}

%token STATUS IS LIGHT NOISE DOOR WINDOWS MOTION LOCK UNLOCK PERCENT THERMOMETER CELSIUS CLEANER SILENT WET NORMAL HEATER TARGET TEMPERATURE

%union

{

int number;

char \*string;

char \*boolean;//(true/false, open/close)

}

%token <number> NUMBER

%token <boolean>STATE1

%token <boolean>STATE2

%token <boolean>STATE3

%token <boolean>STATE4

%token <string> WORD

%%

scripts:

|scripts sensors

|scripts devices

;

devices:

robotCleaner

|

heating

;

heating:

heatSwitch

|

targetSet

;

heatSwitch:

HEATER STATE4

{

outputFile=fopen("output.txt", "a+");

if (strcmp($2,"on") == 0)

fprintf(outputFile,"Обогреватель включен\n");

else {

fprintf(outputFile,"Обогреватель выключен\n");

}

strcpy(devices.heater,$2);

close(outputFile);

}

;

targetSet:

HEATER TARGET TEMPERATURE NUMBER

{

outputFile=fopen("output.txt", "a+");

fprintf (outputFile,"Температура обогревателя %d°C\n",$4);

devices.theater=$4;

close(outputFile);

}

;

robotCleaner:

robot\_cleaner\_switch

|

set\_mode\_for\_robot\_cleaner

;

robot\_cleaner\_switch:

CLEANER STATE4

{

outputFile=fopen("output.txt", "a+");

if (strcmp($2,"on") == 0)

fprintf(outputFile,"Робот-пылесос включен\n");

else

fprintf(outputFile,"Робот-пылесос выключен\n");

devices.cleaner=$2;

close(outputFile);

}

set\_mode\_for\_robot\_cleaner:

silent

|

wet

|

normal

;

silent:

CLEANER SILENT

{

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"Выбран тихий режим уборки для робота-пылесоса\n");

devices.cleaner="silent";

close(outputFile);

}

wet:

CLEANER WET

{

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"Выбран влажный режим уборки для робота-пылесоса\n");

devices.cleaner="wet";

close(outputFile);

}

normal:

CLEANER NORMAL

{

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"Выбран обычный режим уборки для робота-пылесоса\n");

devices.cleaner="normal";

close(outputFile);

}

sensors:

noise

|

light

|

motion

|

windows

|

temperature

|

lockTheDoor

;

lockTheDoor: DOOR IS STATE3

{

home.ttime = time(NULL);

char temp [50];

if (strcmp($3,"unlock") == 0)

strcpy(temp,"Входная дверь открыта");

else

strcpy(temp,"Входная дверь закрыта");

strcpy(home.door,temp);

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"%s, изменено:%s",home.door,ctime(&home.ttime));

close(outputFile);

}

temperature: THERMOMETER IS NUMBER CELSIUS

{

home.thermometer = $3;

home.ttime = time(NULL);

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"Температура в доме:%d°C, изменено:%s",home.thermometer, ctime(&home.ttime) );

close(outputFile);

}

;

windows: WINDOWS IS STATE1

{

home.ttime = time(NULL);

char temp [50];

if (strcmp($3,"open") == 0)

strcpy(temp,"Окна открыты");

else

strcpy(temp,"Окна закрыты");

strcpy(home.windows,temp);

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"%s, изменено:%s",home.windows,ctime(&home.ttime));

close(outputFile);

}

;

light: LIGHT IS NUMBER PERCENT

{

home.light = $3;

home.ttime = time(NULL);

//printf("\nСвет: %d%\n",home.light);

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"Уровень освещённости:%d%, изменено:%s",home.light, ctime(&home.ttime) );

close(outputFile);

}

;

noise: NOISE IS NUMBER PERCENT

{

home.ttime = time(NULL);

home.noise = $3;

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"Уровень шума:%d%, изменено:%s",home.noise, ctime(&home.ttime));

close(outputFile);

}

;

motion: MOTION IS STATE2

{

home.ttime = time(NULL);

char temp [50];

if (strcmp($3,"true") == 0)

strcpy(temp,"Замечено движение");

else

strcpy(temp,"Движения не обнаружены");

strcpy(home.motion,temp);

outputFile=fopen("output.txt", "a+");

fprintf(outputFile,"%s, изменено:%s",home.motion, ctime(&home.ttime));

close(outputFile);

}

;

%%

void yyerror (const char \*str)

{

fprintf(stderr, "ошибка: %s\n",str);

}

int yywrap()

{

return 1;

}

void main()

{

FILE \*fileConf = fopen("config.txt", "r");

yyin=fileConf;

yyparse();

fclose(fileConf);

outputFile=fopen("output.txt", "a");

fprintf(outputFile,"------Активные сценарии работы дома------\n");

if ( (strcmp(home.motion,"Движения не обнаружены") == 0) & (home.light < 10) & (home.noise < 10) )

fprintf(outputFile,"Запущен сценарий работы дома: сон\n");

if ((strcmp(devices.cleaner, "wet") == 0) || (strcmp(devices.cleaner, "normal") == 0))

fprintf(outputFile,"\tРобот-пылесос выключен\n");

if (home.thermometer < 20)

{

fprintf(outputFile,"Запущен сценарий работы дома: отопление\n");

strcpy(home.windows, "Окна закрыты");

fprintf(outputFile,"\t%s\n",home.windows);

devices.theater=30;

devices.heater="Обогреватель включен";

fprintf(outputFile,"\t%s.Температура обогревателя %d°C\n",devices.heater,devices.theater);

}

if ( (strcmp(home.motion,"Движения не обнаружены") == 0) & (strcmp(home.door,"Входная дверь закрыта") == 0) )

fprintf(outputFile,"Запущен сценарий работы дома: дом пуст\n");

fprintf(outputFile,"------Изменения------\n");

close(outputFile);

}

Исходный код лексического анализатора Lex:

%{

#include <stdio.h>

#include <string.h>

#include "y.tab.h" //КАЖДЫЙ РАЗ ОБЩРАЩЯЯСЬ К yylval НЕОБХОДИМО УКАЗЫВАТЬ ТИП ДАННЫХ(yylval.string,number)

%}

%%

unlock|lock {

yylval.boolean=strdup(yytext);

return STATE3;

}

temperature return TEMPERATURE;

target return TARGET;

heater return HEATER;

light return LIGHT;

noise return NOISE ;

door return DOOR ;

windows return WINDOWS ;

motion return MOTION ;

°C return CELSIUS;

% return PERCENT;

cleaner return CLEANER;

silent return SILENT;

wet return WET;

normal return NORMAL;

: return IS;

thermometer return THERMOMETER;

[0-9]+ {

yylval.number=atoi(yytext);

return NUMBER;

}

open|close {

yylval.boolean=strdup(yytext);

return STATE1;

}

true|false {

yylval.boolean=strdup(yytext);

return STATE2;

}

on|off {

yylval.boolean=strdup(yytext);

return STATE4;

}

[a-z]+ {

yylval.string=strdup(yytext);

return WORD;

}

\n /\*игнорим символ конца строки \*/;

[ \t]+ /\*игнорим пробелы и символы табуляции \*/;

%%

# **Список литературы:**

1. Обзор современных языков программирования, ориентированных на решение вычислительных задач / П. А. Акимов, А. М. Белостоцкий, Т. Б. Кайтуков [и др.] // Вопросы прикладной математики и вычислительной механики : Сборник трудов № 20. – Москва : Издательство АСВ, 2017. – С. 54-69.
2. Никифоров, А. Ю. Синтез синтаксисов в создании специализированных языков программирования / А. Ю. Никифоров, В. А. Русаков // Программные продукты и системы. – 2011. – № 4. – С. 20.
3. Ханфенова, С. А. Сравнительный анализ и характеристика популярных языков программирования / С. А. Ханфенова // Молодежь. Наука. образование : материалы республиканской научной студенческой сессии, Карачаевск, 18–20 мая 2020 года. – Карачаевск: Карачаево-Черкесский государственный университет им. У.Д. Алиева, 2020. – С. 339-342.
4. Сибилева, Н. С. Некоторые аспекты реализации лексического анализатора как инструмента интерактивного человеко-машинного взаимодействия / Н. С. Сибилева, А. В. Охотниченко // Теплотехника и информатика в образовании, науке и производстве : Сборник докладов VII Всероссийской научно-практической конференции студентов, аспирантов и молодых учёных (TИМ'2018) с международным участием, Екатеринбург, 17–18 мая 2018 года / Редакционная коллегия: Спирин Н. А., Лавров В. В., Воронов Г. В., Гольцев В. А, Гурин И. А., Казяев М. Д., Киселев Е. В., Куделин С. П., Лошкарев Н. Б., Матюхин В. И., Носков В. Ю., Швыдкий В. С., Ярошенко Ю. Г.. – Екатеринбург: Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, 2018. – С. 512-517.
5. Лаврова, Ю. К. Методика разработки кодогенерирующей части трансляторов на примере семейства трансляторов с Алгола 68 : специальность 05.13.11 "Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей" : автореферат диссертации на соискание ученой степени кандидата физико-математических наук / Лаврова Юлия Кимовна. – Санкт-Петербург, 1995. – 16 с.
6. Монахов, В. И. Обработка больших данных средствами Python на примере анализа репозиториев github / В. И. Монахов, И. И. Плынский // Современные технологии хранения, обработки и анализа больших данных : Сборник научных трудов кафедры автоматизированных систем обработки информации и управления. – Москва : Федеральное государственное бюджетное образовательное учреждение высшего образования "Российский государственный университет имени А.Н. Косыгина (Технологии. Дизайн. Искусство)", 2019. – С. 45-49.
7. Vodyanitskaya, L. A. Применение системы "умный дом" в современном мире / L. A. Vodyanitskaya, D. R. Ryabuchin // Информационно-коммуникативная культура: наука и образование : сборник статей конференции, Ростов-на-Дону, 10–11 января 2019 года. – Ростов-на-Дону: Донской государственный технический университет, 2019. – P. 266-268.
8. Новичкова, М. И. Разработка транслятора языка программирования высокого уровня / М. И. Новичкова, Ю. А. Трофимов // Образование и наука в современных условиях. – 2015. – № 3. – С. 213-214.
9. Авдеев, А. С. Применение концепции "RuleCaster" с использованием модели системы "Умный Дом" на основе сценариев / А. С. Авдеев // Проблемы современной науки и образования. – 2015. – № 2(32). – С. 28-34.
10. Аксенов, М. А. Система "умный дом" для коттеджного дома / М. А. Аксенов, В. А. Казарновский // Аллея науки. – 2019. – Т. 2. – № 2(29). – С. 191-194.