Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Программа: «Цифровые технологии и интеллектуальные системы управления»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Ю. Столбов

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА КУРСОВУЮ РАБОТУ***

***по дисциплине***

**«Языки программирования и методы трансляции»**

Сукерт Милена Константиновна ИСТ-22-1б

(фамилия, имя, отчество студента; группа)

1. **Тема курсовой работы**

Язык программирования для управления умным домом

2. **Срок сдачи студентом отчета:**

28.12.2023

3. **Содержание отчета:**

В курсовой работе были изучены грамматика и принципы работы лексического

анализатора flex и синтаксического анализатора bison, а также с их помощью был

разработан язык программирования для управления умным домом ,в котором продумана

логика if else, AST и взаимодействие с ним через root, также

был проведен анализ ограничений и возможностей созданного ЯП.

С.Е. Батин

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка)

Сукерт М.К.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский**

**политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Программа: «Цифровые технологии и интеллектуальные системы управления»

**КУРСОВАЯ РАБОТА**

по дисциплине

**«Языки программирования и методы трансляции»**

Тема: **«Язык управления умным домом»**

Выполнил:

ИСТ-22-1б

студент гр.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Сукерт М.К.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись, дата)*

Принял:

Доцент каф. ВММБ, к.т.н. С.Е. Батин

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь, 2023**

# Введение

Данная курсовая работа направлена на создание языка программирования с помощью инструментов flex и bison. Данные инструменты используются для создания лексического и синтаксического анализаторов.

Работа должна содержать описание грамматики языка, инструменты реализации грамматики (вышеупомянутые flex и bison), исходные коды лексического и синтаксического анализаторов, а также ограничения, связанные с работой языка программирования.

Результатом работы будет являться свой язык программирования для умного дома.

Лексический анализатор

Термин "flex" часто используется для обозначения лексического анализатора, который является частью инструментов компиляторов. Flex (Fast Lexical Analyzer Generator) — это программный инструмент для автоматической генерации лексических анализаторов.

Лексический анализатор преобразует входной текст программы в последовательность лексем (токенов), которые затем обрабатываются синтаксическим анализатором. Flex работает на основе спецификаций в виде регулярных выражений и соответствующих действий. Он генерирует код лексического анализатора на языке программирования C или C++.

Пример использования Flex:

Пользователь определяет правила сопоставления лексем с использованием регулярных выражений.

Flex генерирует соответствующий код лексического анализатора, который может быть интегрирован в компилятор.

Использование инструментов, таких как Flex, позволяет разработчикам упростить процесс создания лексических анализаторов и улучшить поддержку для различных языков программирования.

# Синтаксический анализатор

Bison — это генератор синтаксических анализаторов, который работает в паре с лексическим анализатором Flex. Он широко используется в компиляторах и других инструментах обработки языков программирования. Bison генерирует синтаксические анализаторы на основе заданных пользователем правил грамматики.

Вот основные шаги по созданию синтаксического анализатора с использованием Bison:

**Определение грамматики:** Пользователь определяет грамматику своего языка программирования с использованием формальных правил, которые описывают структуру языка. Грамматика обычно записывается в форме контекстно-свободной грамматики.

**Создание файла с правилами Bison:** В этом файле пользователь использует специальный синтаксис для определения правил грамматики. Эти правила включают в себя определения нетерминалов, терминалов и действий, которые выполняются при распознавании конкретных конструкций.

**Генерация синтаксического анализатора:** Bison обрабатывает файл с правилами и генерирует код синтаксического анализатора на языке программирования C или C++.

**Интеграция с лексическим анализатором:** Полученный синтаксический анализатор интегрируется с лексическим анализатором, который был создан с использованием Flex.

**Разработка и отладка:** После генерации кода разработчик может дополнить его дополнительными функциями и провести отладку.

Пример грамматики для Bison может выглядеть следующим образом:

program: statement\_list

;

statement\_list: statement

| statement\_list statement

;

statement: create\_ast\_statement SEMICOLON

| expression\_statement SEMICOLON

| light\_command SEMICOLON

| conder\_command SEMICOLON

| status\_command SEMICOLON

| set\_temperature\_statement SEMICOLON

| print\_statement SEMICOLON

| if\_else\_statement

| variable\_declaration SEMICOLON

| assignment\_statement SEMICOLON

;

Еще в грамматике yacc описана с использование варианта формы Бекуса-Наура (BNF). Это техника, разработанная Джоном Бэкусом и Питером Науром. (рис. 1)

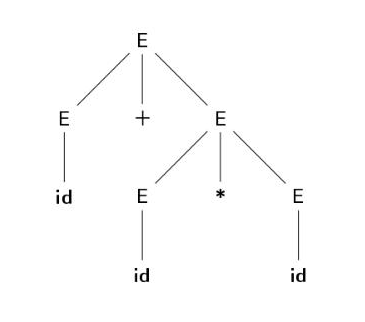


Рис 1. Дерево разбора

Дерево разбора (или синтаксическое дерево) в программировании представляет собой структуру данных, которая отражает синтаксическую структуру входного кода в соответствии с определенной грамматикой. Это дерево служит визуальным представлением иерархии элементов программы, обеспечивая более удобное представление для дальнейшего анализа и обработки.

Процесс построения дерева разбора обычно выполняется с использованием синтаксического анализатора (парсера), который анализирует входные данные (например, исходный код программы) в соответствии с определенной грамматикой языка программирования. Дерево разбора состоит из узлов, представляющих синтаксические элементы кода, и ребер, отображающих связи между этими элементами.

Деревья разбора используются в компиляторах и интерпретаторах для представления синтаксической структуры программы. После построения дерева разбора происходит следующий этап анализа (семантический анализ, оптимизация, генерация кода), который использует информацию, полученную из этой структуры, для дальнейших шагов обработки программы.

# Грамматика языка

В файле [input.txt](https://github.com/MilenaSkrt/smarthome/blob/main/input.txt) записан разработанный псевдокод.

1)

create\_ast "room";

"room".status();

"room".set\_temperature(23);

"room".light\_on();

**if** ("room".temperature == 23) {

"room".light\_off();

"room".set\_temperature(20);

} **else** {

"room".light\_off();

}

"room".status();

2)

create\_ast "room";

"room".status();

"room".set\_temperature(23);

"room".light\_on();

**if** ("room".temperature == 23) {

"room".light\_off();

"room".set\_temperature(20);

} **else** {

"room".light\_off();

}

"room".status();

int A == 3;

print(A \* 5);

После компилирования программы следующими командами:

flex lexer.l

bison -d parser.y

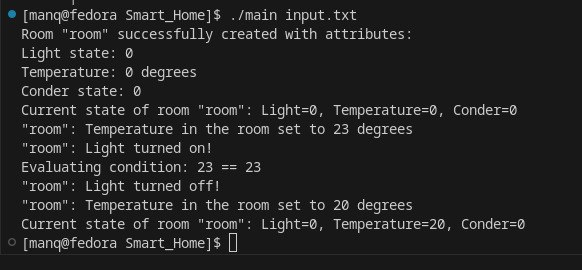
gcc -o main parser.tab.c lex.yy.c struct\_func.c -lm

./main input.txt

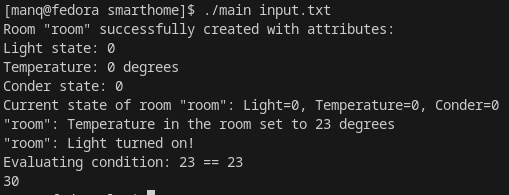
(не обращая внимания на предупреждения)

Мы получаем следующий вывод в консоль:

1)



2)



Реализована работа с таблицей символов и реализованы переменные и операции над ними:

*// Symbol Table*

**typedef** **struct** {

char\* name;

int type; *// 0 for int, 1 for string*

int intValue;

char\* stringValue;

} Symbol;

Symbol symbolTable[100];

int symbolCount = 0;

void addSymbol(char\* name, int type, int intValue, char\* stringValue) {

Symbol symbol;

symbol.name = name;

symbol.type = type;

symbol.intValue = intValue;

symbol.stringValue = stringValue;

symbolTable[symbolCount++] = symbol;

}

Symbol\* findSymbol(char\* name) {

**for** (int i = 0; i < symbolCount; i++) {

**if** (strcmp(symbolTable[i].name, name) == 0) {

**return** &symbolTable[i];

}

}

**return** NULL;

}

assignment\_statement:

ID EQUAL expression {

Symbol\* symbol = findSymbol($1);

**if** (symbol != NULL) {

**if** (symbol->type == 0) {

symbol->intValue = $3;

} **else** {

fprintf(stderr, "Error: Cannot assign integer value to string variable '%s'**\n**", $1);

exit(EXIT\_FAILURE);

}

} **else** {

fprintf(stderr, "Error: Variable '%s' not found**\n**", $1);

exit(EXIT\_FAILURE);

}

}

;

variable\_declaration:

INT\_TYPE ID EQUAL INTEGER {

addSymbol($2, 0, $4, NULL);

}

| STRING\_TYPE ID EQUAL STRING\_VALUE {

addSymbol($2, 1, 0, $4);

}

;

# Анализ ограничений

В данном языке не реализованы переменные, конструкции while и for которые бы значительно расширили возможности языка.

# Заключение

Был создан язык программирования для управления умным домом. Язык поддерживает взаимодействия с устройствами и базовую конструкцию if else.

**Приложение**

Файлы проекта :  
[input.txt](https://github.com/MilenaSkrt/smarthome/blob/main/input.txt)

[parser.y](https://github.com/MilenaSkrt/smarthome/blob/main/parser.y)

[struct\_func.c](https://github.com/MilenaSkrt/smarthome/blob/main/struct_func.c)

[struct\_func.h](https://github.com/MilenaSkrt/smarthome/blob/main/struct_func.h)

[lexer.l](https://github.com/MilenaSkrt/smarthome/blob/main/lexer.l)