|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Лабораторная работа № 1 | | |
| по дисциплине «ЯЗЫКИ ПРОГРАММИРОВАНИЯ И МЕТОДЫ ТРАНСЛЯЦИИ» | | |
| **ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ТАБЛИЦ, ИСПОЛЬЗУЕМЫХ В ТРАНСЛЯТОРЕ** | | |
|  | | |
|  | Бригада 4 | затолоцкая юлия |
| Группа ПМ-91 | Барсукова наталья |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | И.Л. Еланцева |
|  |  |
| Новосибирск, 2022 | | |

1. **Цели и задачи проекта**

Получить представление о видах таблиц, используемых при трансляции программ. Изучить множество операций с таблицами и особенности реализации этих операций для таблиц, используемых на этапе лексического анализа. Реализовать классы таблиц, используемых сканером.

1. **Вид, объем и источник исходных данных;**

Исходными данными для постоянных таблиц являются файлы, где перечислены элементы этих таблиц. В файлах каждый идентификатор начинается с новой строки.

Для переменных таблиц явных исходных данных нет. Они формируются динамически.

1. **Структура таблиц**

*Класс для работы с постоянными таблицами*

*Имя класса:* const\_table

Методы класса:

|  |  |
| --- | --- |
| Имя метода | Описание |
| const\_table(string filename) | Конструктор, генерирует таблицу из файла с именем filename |
| const\_table(); | Конструктор по умолчанию |
| string get(int number); | Возвращает элемент по номеру в таблице |
| int search(string value); | Проверяет наличие элемента в таблице. Возвращает номер элемента в таблице, если элемент есть в таблице, иначе -1. |

*Класс для работы с переменными таблицами*

*Имя класса:* variable\_table

Структура данных представляет хэш-таблицу с цепочками

Методы класса:

|  |  |
| --- | --- |
| Имя метода | Описание |
| variable\_table(); | Конструктор, генерирует пустую хэш-таблицу размером 32. |
| void settype(int type, int hash, int i); | Устанавливает тип type элементу с хэшем hash и индексом в цепочке i. |
| pair <int, int> set(identifier id); | Добавляет элемент в таблицу и возвращает хэш и номер в цепочке |
| pair <int, int> setname(string name); | Добавляет элемент по имени name в таблицу и возвращает хэш и номер в цепочке |
| void setvalue(string name, string value); | Устанавливает значение value элементу с именем name |
| pair <int, int> search(string name); | Ищет элемент с именем name и возвращает хэш и номер в цепочке |
| pair <int, int> search(identifier id); | Ищет элемент id и возвращает хэш и номер в цепочке |
| int gettype( int hash, int i); | Возвращает тип элемента с хэшем hash и номером в цепочке i |
| string getvalue(int hash, int i); | Возвращает значение элемента с хэшем hash и номером в цепочке i |
| int calc\_hash(string name); | Возвращает хэш по имени элемента |

*Класс для работы со всеми таблицами*

*Имя класса:* Tables

Хранит в себе таблицы

const\_table words, operations, separators, alphabet, digits;

variable\_table constants, variables;

Методы класса:

|  |  |
| --- | --- |
| Имя метода | Описание |
| Tables(); | Конструктор, создает все таблицы с помощью конструкторов |
| trio find(string c); | Ищет элемент во всех таблицах, возвращает номер таблицы, номер хэша и номер в цепочке |

1. **Тексты программ**

*Tables.h*

#pragma once

#include <string>

#include <vector>

#include <iostream>

#include <fstream>

using namespace std;

struct identifier

{

string name;

int type;

string value;

identifier(string name, int type, string value);

identifier();

};

class const\_table

{

public:

vector<string> table;

const\_table(string filename);// читать из файла

const\_table();

string get(int number);

int search(string value);

};

struct trio

{

int table;

int ind;

int sind;

trio(int t, int i, int si = 0);

trio(int t, pair <int, int> par\_index);

};

/\*---------------------------------\*/

class variable\_table

{

public:

vector<vector<identifier>> table;

variable\_table();

void settype(int type, int hash, int i);

pair <int, int> set(identifier id);

pair <int, int> setname(string name);

void setvalue(string name, string value);

pair <int, int> search(string name);

pair <int, int> search(identifier id);

int gettype( int hash, int i);

string getvalue(int hash, int i);

int calc\_hash(string name);

};

class Tables

{

public:

const\_table words, operations, separators, alphabet, digits;

variable\_table constants, variables;

Tables();

trio find(string c);

};

*Tables.cpp*

#include "Tables.h"

const\_table::const\_table(string filename)

{

ifstream rfile (filename);

if (rfile.is\_open()) {

string s;

while (!rfile.eof()) { // keep reading until end-of-file

getline(rfile, s);

table.push\_back(s);

}

}

rfile.close();

}

const\_table::const\_table()

{

}

string const\_table::get(int number)

{

return table[number];

}

int const\_table::search(string value)

{

for (int i = 0; i < table.size(); i++)

{

if (value == table[i])

{

return i;

}

}

return -1;

}

/\*------------------------------------------\*/

variable\_table::variable\_table()

{

table.resize(32);

}

void variable\_table::settype(int type, int hash, int i)

{

table[hash][i].type = type;

}

pair <int, int> variable\_table::set(identifier id)

{

unsigned int hash = calc\_hash(id.name) % table.size();

pair <int, int> index = search(id.name);

if (index.first == -1)

{

table[hash].push\_back(id);

return { hash, table[hash].size() - 1 };

}

return { index.first, index.second };

}

pair <int, int> variable\_table::setname(string name)

{

unsigned int hash = calc\_hash(name) % table.size();

pair <int, int> index = search(name);

identifier id(name, 0, "");

if (index.first == -1)

{

table[hash].push\_back(id);

return { hash, table[hash].size() - 1 };

}

return { index.first, index.second };

}

void variable\_table::setvalue(string name, string value)

{

pair <int, int> index = search(name);

if (index.first != -1)

{

table[index.first][index.second].value=value;

}

}

pair <int, int> variable\_table::search(string name)

{

int hash = calc\_hash(name) % table.size();

if (table[hash].size() == 0)

{

return { -1, 0 };

}

for (int i = 0; i < table[hash].size(); ++i)

{

if (table[hash][i].name == name)

{

return { hash, i };

}

}

return { -1, 0 };

}

pair<int, int> variable\_table::search(identifier id)

{

return search(id.name);

}

int variable\_table::gettype( int hash, int i)

{

return table[hash][i].type;

}

string variable\_table::getvalue( int hash, int i)

{

return table[hash][i].value;

}

int variable\_table::calc\_hash(string name)

{

unsigned int b = 378551;

unsigned int a = 63689;

unsigned int hash = 0;

for (unsigned int i = 0; i < name.length(); i++)

{

hash = hash \* a + name[i];

a = a \* b;

}

return (hash & 0x7FFFFFFF);

}

/\*------------------------------------------\*/

identifier::identifier(string nam, int typ, string valu)

{

name = nam;

type = typ;

value = valu;

}

identifier::identifier()

{

name = "";

type = 0;

value = "0";

}

/\*------------------------------------------\*/

Tables::Tables()

{

words = const\_table("words.txt");

operations= const\_table("operations.txt");

separators = const\_table("separators.txt");

alphabet = const\_table("alphabet.txt");

digits = const\_table("digits.txt");

variables = variable\_table();

constants = variable\_table();

}

trio Tables::find(string c)

{

int p;

p = separators.search(c);

if (p != -1)

return trio(0, p);

p = words.search(c);

if ( p!= -1)

return trio(1, p);

p = operations.search(c);

if ( p!= -1)

return trio(2, p);

pair <int, int> pa;

pa = variables.search(c);

if (pa.first != -1)

return trio(3, pa);

pa = constants.search(c);

if (pa.first != -1)

return trio(4, pa);

return trio(-1, 0);

}

/\*------------------------------------------\*/

trio::trio(int t, int i, int si)

{

table = t;

ind = i;

sind = si;

}

trio::trio(int t, pair<int, int> par\_index)

{

table = t;

ind = par\_index.first;

sind = par\_index.second;

}

*Lab1.cpp*

#include <iostream>

#include "Tables.h"

int main()

{

Tables tables = Tables();

string c = tables.separators.get(2);

int i = tables.separators.search("{");

trio id = tables.find("{");

id = tables.find("a");

id = tables.find("if");

id = tables.find("for");

string v = "hello";

identifier var = identifier("a", 0,v );

tables.variables.set(var);

var = identifier("a", 0, v);

tables.variables.search("a");

id = tables.find("a");

tables.variables.setvalue("a", "bi");

tables.variables.getvalue(1, 0);

tables.variables.settype(2, 1,0);

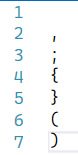
tables.variables.gettype(1, 0);

}

1. **Тестовые примеры**

*Тест для постоянных таблиц*

Исходный файл с именем “separators.txt”:



|  |  |  |
| --- | --- | --- |
| Шаг | Описание | Результат |
| separators = const\_table("separators.txt"); | Создание таблицы из файла |  |
| string c = tables.separators.get(2); | Вернуть элемент под номером 2 в таблице | c = ; |
| int i = tables.separators.search("{"); | Вернуть позицию элемента '{' в таблице, если найден | I = 3 |
| id = tables.find("if"); | Вернуть -1, если не найден | Id = -1 |

*Тест для переменных таблиц*

|  |  |  |
| --- | --- | --- |
| Шаг | Описание | Результат |
| variables = variable\_table(); | Создание пустой таблицы variables |  |
| identifier var = identifier("a", 0, "hello" ); tables.variables.set(var); | Добавить идентификатор в таблицу. |  |
| tables.variables.search("a"); | Поиск элемента по имени | (1, 0) |
| tables.variables.setvalue("a", "bi"); | Установить значение "bi" для переменной a |  |
| tables.variables.getvalue(1, 0); | Возвратить значение переменной а | bi |
| tables.variables.settype(1, 1, 0); | Установить тип 1 для переменной а |  |
| tables.variables.gettype(1, 0); | Вернуть тип переменной а | 1 |

*Тест для всех таблиц*

|  |  |  |
| --- | --- | --- |
| Шаг | Описание | Результат |
| Tables tables = Tables(); | Создание всех таблиц |  |
| trio id = tables.find("{"); | Найти элемент "{" и вернуть индексы | (0,3,0) |
| id = tables.find("a"); | Найти элемент "a" и вернуть индексы | (1,0,0) |
| id = tables.find("if"); | Найти элемент "if" и вернуть индексы | (-1,0,0) |
| id = tables.find("for"); | Найти элемент "for" и вернуть индексы | (1,2,0) |