МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6-05-0612-01 Программная инженерия

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора LPA-2024»

Выполнил студент Ленкевич Павел Андреевич

(Ф.И.О.)

Руководитель проекта Волчек Дарья Ивановна

Заведующий кафедрой к.т.н., доц. Смелов В.В.

Консультанты Волчек Дарья Ивановна

Нормоконтролер Волчек Дарья Ивановна

Курсовой проект защищен с оценкой

Минск 2024

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6-05-0612-01 Программная инженерия

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024г.

**ЗАДАНИЕ**

**к курсовому проектированию по дисциплине**

«Конструирование программного обеспечения»

Специальность: Программная инженерия Группа: 7

Студент: Ленкевич Павел Андреевич

(фамилия, имя, отчество)

**1. Тема проекта** Разработка компилятора LPA-2024

**2. Срок сдачи студентом законченного проекта:** \_\_\_ декабря 2024 г.

**3. Исходные данные к проекту:**

Разработка компилятора осуществляется на языке C++ (стандартизации International Standard ISO/IEC 14882:2020(E) Programming Language C++ 20) в среде разработки Visual Studio Community 2022 Preview 6.0. Операционная система, под которой происходит разработка Windows 10 (64-bit). Типы данных: целочисленный (2 байта), строковый. Представление литералов: целые, представление – восьмеричное и десятичное; строковые и/или символьные – произвольно. Арифметические операции: сложение, вычитание, умножение, деление нацело, остаток от деления. Операторы вывода в стандартный поток: write, writeline. Функции стандартной библиотеки: копирование строк; вычисление длины строки. Инструкции: оператор цикла.

**4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):**

Введение

1) Спецификация языка программирования

2) Структура транслятора

3) Разработка лексического анализатора

4) Разработка синтаксического анализатора

5) Разработка семантического анализатора

6) Вычисление выражений

7) Генерация кода

8) Тестирование транслятора (Разработка и тестирование интерпретатора)

9) Заключение

10) Литература

Приложения

Графическая часть

**5. Перечень графического материала (с точным указанием обязательных чертежей)**

1) Граф дерева разбора

**6. Консультанты по проекту с указанием относящихся к ним разделов проекта**

|  |  |
| --- | --- |
| Раздел | Консультант |
| Разработка лексического анализатора | Волчек Д. И. |
| Разработка синтаксического и семантического анализатора. | Волчек Д. И. |
| Генерация кода. | Волчек Д. И. |
| Разработка тестовых примеров. | Волчек Д. И. |
| Оформление пояснительной записки к курсовому проект. | Волчек Д. И. |

**7. Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Спецификация специализированного языка LPA-2024 | 25.09.2024 |  |
| 2 | Разработка лексического анализатора | 30.09.2024 |  |
| 3 | Разработка синтаксического анализатора | 11.10.2024 |  |
| 4 | Разработка семантического анализатора | 25.10.2024 |  |
| 5 | Генерация кода | 08.11.2024 |  |
| 6 | Тестирование компилятора | 22.11.2024 |  |
| 7 | Оформление пояснительной записки к курсовому проект | 06.12.2024 |  |
| 8 | Сдача проекта | 16.12.2024 |  |

**8. Дата выдачи задания 18.09.2024**

Руководитель Волчек Д. И.

(подпись)

Задание принял к исполнению Ленкевич П. А.

(подпись студента)

**Содержание**

|  |  |
| --- | --- |
| Введение | 5 |
| 1. Постановка задачи | 6 |
| 1.1. Обзор аналогичных решений | 6 |
| 1.2. Техническое задание | 8 |
| 1.3. Выбор средств реализации программного продукта | 9 |
| 1.4. Вывод | 9 |
| 2. Проектирование страниц веб-сайта | 10 |
| 2.1. Выбор способа верстки | 10 |
| 2.2. Выбор стилевого оформления | 10 |
| 2.3. Выбор шрифтового оформления | 10 |
| 2.4. Разработка логотипа | 10 |
| 2.5. Разработка пользовательских элементов | 11 |
| 2.6. Разработка спецэффектов | 13 |
| 2.7. Выводы | 14 |
| 3. Реализация структуры веб-сайта | 15 |
| 3.1. Структура HTML-документа | 15 |
| 3.2. Добавление таблиц стилей SCSS и CSS | 16 |
| 3.3. Использование стандартов XML (SVG) | 16 |
| 3.4. Управление элементами DOM | 17 |
| 3.5. Выводы | 18 |
| 4. Тестирование веб-сайта | 19 |
| 4.1. Адаптивный дизайн веб-сайта | 19 |
| 4.2. Кроссбраузерность веб-сайта | 21 |
| 4.3. Руководство пользователя | 22 |
| 4.4. Выводы | 23 |
| Заключение | 24 |
| Список использованных литературных источников | 25 |
| Приложение А Прототипы веб-страниц | 26 |
| Приложение Б Макет структуры веб-сайта | 33 |
| Приложение В Листинг НТML-документа | 40 |
| Приложение Г Листинг SCSS и CSS | 56 |
| Приложение Д Листинг XML-файлов | 90 |
| Приложение Е Листинг SVG | 94 |
| Приложение Ж Листинг JavaScript | 97 |

# **Введение**

Целью курсового проекта по дисциплине «Конструирование программного обеспечения» является написание спецификации и разработка компилятора для собственного языка программирования LPA-2024, ориентированного на выполнение простейших арифметических операций в консольном режиме. Компиляция кода языка LPA-2024 будет производиться в язык ассемблер.

Этапы разработки компилятора для языка LPA-2024:

– разработка спецификации языка программирования;

– разработка структуры транслятора;

– разработка лексического анализатора;

– разработка лексического анализатора;

– разработка синтаксического анализатора;

– разработка семантического анализатора;

– обработка выражений;

– генерация кода на язык Assembler;

– тестирование транслятора.

Информация о каждом этапе разработки компилятора приведена в соответствующих разделах пояснительной записки.

В первом разделе приведена спецификация языка программирования.

Во втором разделе описана структура транслятора.

В третьем разделе описаны принцип работ и этапы разработки лексического анализатора, определены разрешенные символы и ключевые слова языка программирования.

В четвертом разделе описан принцип работы синтаксического анализатора.

В пятом разделе описаны принцип работы и основные функции семантического анализатора.

В шестом разделе описаны выражения, допускаемые языком, форма, принципы построения и вычисления выражений.

В седьмом разделе описан процесс генерации кода.

В восьмом разделе приведены примеры тестирования транслятора.

Язык программирования LPA-2024 предназначен для работы с консолью, выполнения простейших арифметических операций.

# **Спецификация языка программирования**

## **Характеристика языка программирования**

LPA-2024 – это компилируемый, строго типизированный, универсальный язык программирования, ориентированный на выполнение простейших арифметических операций в консольном режиме.

## **Определение алфавита языка программирования**

Алфавит языка LPA-2024 основан на кодировке Windows-1251, изображённой на рисунке 1.1.



Рисунок 1.1 – Кодировка Windows-1251

## **Применяемые сепараторы**

Символы-сепараторы служат в качестве разделителей цепочек языка во время обработки исходного текста программы с целью разделения на токены.

Сепараторы, применяемые в языке LPA-2024, приведены в таблице 1.1.

Таблица 1.1 Символы-сепараторы

|  |  |
| --- | --- |
| Разделители | Назначение |
| «пробел», «табуляция», «переход на новую строку» | Разделяют входные лексемы |
| +, -, \*, /, % | Арифметические операторы. Используются в арифметических операторах |
| = | Оператор присваивания. Используется для присваивания значения переменной |
| <, >, <=, >=, !=, == | Условные операторы. Используются в условии цикла |
| ( ) | Блок параметров функции, так же указывает на приоритет в арифметических операциях |
| , | Разделяет параметры функции |
| { } | Ограничивает программные конструкции |
| ; | Признак конца инструкции языка |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования LPA-2024 используется кодировка Windows-1251.

## **1.5 Типы данных**

В языке LPA-2024 поддерживается 2 типа данных: целочисленный (2 байта) и строковый. Пользовательские типы данных не поддерживается. Описание типов данных представлено в таблице 1.2.

Таблица 1.2 – Типы данных языка программирования LPA-2024

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| Целочисленный  (int) | В памяти занимает 2 байта.  Максимальное значение: 32767.  Минимальное значение: -32768.  Значение по умолчанию: 0.  Операции: сложение (+), вычитание (-), умножение (\*), деление нацело (/), остаток от деления (%), присваивание (=), а также условия для операторов цикла: больше (>), меньше (<), проверка на неравенство (!). |
| Строковый  (str) | В памяти занимает n + 1 байт, где n – количество символов + символ конца строки.  Максимальное количество символов в строке: 256.  Значение по умолчанию: строка нулевой длины «».  Операции: присваивание идентификатору значения другого строкового идентификатора, строкового литерала или значение строковой функции, использование библиотечных функций. |

## **Преобразование типов данных**

Преобразования типов данных в языке LPA-2024 не предусмотрено.

## **Идентификаторы**

Идентификаторы используются для именования переменных, функций. Идентификатор должен начинаться с строчной буквы латинского алфавита. После первого символа допускается использование строчных букв латинского алфавита и цифр. Идентификаторы не могут совпадать с ключевыми словами языка.

Пример корректных идентификаторов: yasha1, y002asha, yasha2.

Пример некорректных идентификаторов: 1yasha (начинается с цифры), Y2asha (содержит прописную букву), #yaSha2 (содержит недопустимый символ и прописную букву).

## **Литералы**

Литерал представляет собой запись в исходном коде программы, которая обозначает фиксированное значение. В языке программирования LPA-2024 поддерживается два типа литералов: целочисленные и строковые. Целочисленные литералы могут быть представлены в двух системах счисления: восьмеричной и десятеричной.

Описание литералов приведено в таблице 1.3.

Таблица 1.3 – Литералы языка LPA-2024

|  |  |
| --- | --- |
| Тип литерала | Характеристика |
| Целочисленный | Восьмеричный:  Последовательность восьмеричных цифр 0..7 с предшествующим знаком минус или без него, начинающаяся с символа «0» (признак восьмеричного целочисленного литерала).  Десятичный: Последовательность десятичных цифр 0..9 с предшествующим знаком минус или без него.  Допустимый диапазон значений:  От -32768 до 32767 в десятичной системе исчисления. |
| Строковый | Набор, состоящий из символов русского и латинского алфавитов, десятичных цифр и специальных символов, заключённый в двойные кавычки. Допустимый диапазон значений:  От 0 до 255 символов. |

Примеры корректных литералов: “str154str”, 056, -056, 56, -56, 07.

Примеры некорректных литералов: str154str (не заключён в кавычки), o56 (недопустимый символ), 09 и -08 (восьмеричный литерал может принимать только цифры от 0 до 7).

## **Объявление данных**

В языке LPA-2024 для объявления переменных используются ключевые слова int и str. Ключевое слово int обозначает целочисленные переменные, а str – строковые. После ключевого слова указывается имя идентификатора. Инициализация переменной допустима прямо в объявлении.

int <имя идентификатора>;

int <имя идентификатора> = <литерал>;

str <имя идентификатора>;

str <имя идентификатора> = <литерал>;

Переменные могут быть объявлены в блоке main, в блоке функции или внутри цикла. Область видимости идентификаторов определяется границами блока кода, заключённого в фигурные скобки { }.

## **1.10 Инициализация данных**

В языке программирования LPA-2024 поддерживаются два способа инициализации переменных:

– Инициализация в месте объявления – задаётся непосредственно при объявлении переменной:

int <имя идентификатора> = <литерал>;

str <имя идентификатора> = <литерал>;

– Инициализация после объявления – выполняется присваиванием значения уже объявленной переменной:

<имя идентификатора> = <литерал>;

При объявлении переменных без явного указания значения инициализация выполняется автоматически значениями по умолчанию:

* для типа int – 0
* для типа str – пустая строка «».

Примеры инициализации в месте объявления:

* int count = 11;
* str name = “LPA-2024”;

Примеры инициализации после объявления:

* int total;

total = 33;

* str username;

username = “Kapetto”;

Инструкции языка LPA-2024 приведены в таблице 1.4.

Таблица 1.4 – Инструкции языка LPA-2024

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Объявление переменной | <тип данных> <имя идентификатора>; |
| Объявление переменной с явной инициализацией | <тип данных> <имя идентификатора> = <значение>|<выражение>;  Значение – литерал, идентификатор, вызов функции соответствующего типа данных. |

Продолжение таблицы 1.4

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Присваивание | <имя идентификатора> = <значение>; |
| Объявление функции | <тип данных> <имя идентификатора> (<тип данных> <имя идентификатора>, …) {  /тело функции/  return <имя идентификатора/литерал>.  }; |
| Возврат из функции | Для функций, возвращающих значение:  return <имя идентификатора/литерал>; |
| Вызов функции | <имя идентификатора> (<имя идентификатора>, …); |
| Вывод данных | write <литерал/имя идентификатора>;  writeline <литерал/имя идентификатора>; |

## **1.12 Операции языка**

В языке LPA-2024 предусмотрены два типа операций: арифметические и логические.

Наибольшая приоритетность у операций умножения, деления и деления с остатком, затем идут операции сложения и вычитания. Можно задать самый высокий приоритет, поместив операции в скобки.

Все логические операции имеют равный приоритет.

Описание операций языка LPA-2024 приведено в таблице 1.5

Таблица 1.5 – Операции языка LPA-2024

|  |  |
| --- | --- |
| Тип оператора | Оператор |
| Арифметические операции | \* – умножение  / – деление  % – остаток от деления  + – сложение  - – вычитание  = – присваивание |
| Логические операции | > – больше  < – меньше  >= – больше или равно  <= – меньше или равно  == – проверка на равенство  != – проверка на неравенство |

## **1.13 Выражения и их вычисления**

Выражение языка программирования LPA-2024 представляет собой совокупность переменных, литералов, вызовов функций, знаков операций, скобок, которая может быть вычислена в соответствии с синтаксисом языка.

Правила составления выражений:

* Выражение записывается в строку без переносов;
* В выражении могут присутствовать только операнды одинакового типа;
* В выражении могут использоваться функции. Как стандартные, так и пользовательские;
* В выражении не могут идти подряд два оператора;
* Допускается использование круглых скобок для смены приоритета операций.

В арифметических выражениях допускаются только операнды целочисленного типа.

Перед генерацией кода выражения приводятся к ПОЛИЗ для более удобного вычисления на языке ассемблера.

## **Конструкции языка**

Программа на языке LPA-2024 оформляется в виде функций пользователя и главной функции. При составлении функций рекомендуется выделять блоки и фрагменты и применять отступы для улучшения читаемости кода.

Программные конструкции языка представлены в таблице 1.6.

Таблица 1.6 – Программные конструкции языка LPA-2024

|  |  |
| --- | --- |
| Конструкция | Описание |
| Главная функция | main  {  …  }; |

Продолжение таблицы 1.6

|  |  |
| --- | --- |
| Конструкция | Описание |
| Пользовательская функция | <тип данных возвращаемого значения> <имя идентификатора> (<тип данных параметра> <имя идентификатора параметра>, …)  {  …  return <имя идентификатора переменной/литерал>;  };  Максимальное количество параметров: 8. |
| Цикл | cicle (<имя идентификатора переменной/литерал> <условный оператор> <имя идентификатора переменной/литерал>)  {  …  } |

## **Область видимости идентификаторов**

Область видимости: сверху вниз (как в C++). Переменные, объявленные в одной функции, недоступны в другой. Все объявления и операции с переменными происходят внутри блока. Каждая переменная или параметр функции получают префикс – название функции, внутри которой они находятся.

Все идентификаторы являются локальными и обязаны быть объявлены внутри какой-либо функции. Глобальных переменных нет. Параметры видны только в функции, внутри которой объявлены.

## **Семантические проверки**

Семантическим анализатором языка LPA-2024 предусмотрены следующие проверки:

* Наличие блока main – точки входа в программу;
* Единственность точки входа;
* Использование идентификаторов до их объявления;
* Переопределение идентификаторов;

– Соответствие параметров, передаваемых в функцию, с параметрами в объявлении функции;

* Соответствие типа возвращаемого значения с типом функции;
* Соответствие типов в выражениях;
* Превышение размера целочисленных и строковых литералов;
* Превышение длины лексемы;

– Соответствие операторов типам данных, для работы с которыми они предназначены.

## **Распределение оперативной памяти на этапе выполнения**

Транслированный код использует две области памяти. В сегмент констант записываются все литералы языка. В сегмент данных записываются все переменные и параметры функций. Локальная область видимости в исходном коде определяется за счёт использования правил именования идентификаторов и регулируется их префиксами, что и обуславливает их локальность на уровне исходного кода несмотря на то, что в оттранслированным в язык ассемблера коде переменные имеют глобальную область видимости.

## **Стандартная библиотека и её состав**

В языке LPA-2024 присутствует стандартная библиотека, которая подключается автоматически при трансляции исходного кода в язык ассемблера. Содержание библиотеки и описание функций представлено в таблице 1.7.

Таблица 1.7 – Стандартная библиотека языка LPA-2024

|  |  |
| --- | --- |
| Функция | Описание |
| str strcopy (str parm1, str parm2); | Строковая функция, выполняет копирование строки parm2 в parm1. Возвращает parm1. |
| int strlength (str parm1); | Целочисленная функция. Вычисляет и возвращает длину строки parm1. |
| int strtransint (str parm1); | Целочисленная функция. Преобразует строку parm1 в число. Строка не должна состоять из буквенных или специальных символов. Необходима для работы функции strlength. Возвращает число. |

Стандартная библиотека написана на языке C++, подключается на этапе компоновки. Вызовы стандартных функций доступны там же, где и вызов пользовательских функций.

## **Ввод и вывод данных**