МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора SVY-2020»

Выполнил студент Савченко Владислав Юрьевич

(Ф.И.О.)

Руководитель проекта пр.ст. Пахолко Алена Степановна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Пацей Н.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты пр.ст. Пахолко Алена Степановна

(учен. степень, звание, должность, подпись, Ф.И.О.)

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер пр.ст. Пахолко Алена Степановна

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2020

**Содержание**

[**Введение** 1](#_Toc501385915)

[**Глава 1. Спецификация языка программирования** 6](#_Toc501385916)

[**1.1. Характеристика языка программирования** 6](#_Toc501385917)

[**1.2.** **Алфавит языка** 6](#_Toc501385918)

[**1.3. Символы сепараторы** 7](#_Toc501385919)

[**1.4. Применяемые кодировки** 7](#_Toc501385920)

[**1.5. Типы данных** 7](#_Toc501385921)

[**1.6. Преобразование типов данных** 8](#_Toc501385922)

[**1.7. Идентификаторы** 8](#_Toc501385923)

[**1.8. Литералы** 8](#_Toc501385924)

[**1.9. Область видимости идентификаторов** 8](#_Toc501385925)

[**1.10. Инициализация данных** 9](#_Toc501385926)

[**1.11.** **Инструкции языка** 9](#_Toc501385927)

[**1.12. Операции языка** 10](#_Toc501385928)

[**1.13. Выражения и их вычисления** 10](#_Toc501385929)

[**1.14. Программные конструкции языка** 10](#_Toc501385930)

[**1.15.** **Область видимости** 11](#_Toc501385931)

[**1.16. Семантические проверки** 11](#_Toc501385932)

[**1.17.** **Распределение оперативной памяти на этапе выполнения** 11](#_Toc501385933)

[**1.18. Стандартная библиотека и её состав** 11](#_Toc501385934)

[**1.19.** **Ввод и вывод данных** 12](#_Toc501385935)

[**1.20. Точка входа** 12](#_Toc501385936)

[**1.21.** **Препроцессор** 12](#_Toc501385937)

[**1.22. Соглашения о вызовах** 12](#_Toc501385938)

[**1.23.** **Объектный код** 12](#_Toc501385939)

[**1.24. Классификация сообщений транслятора** 12](#_Toc501385940)

[**1.25.** **Контрольный пример** 13](#_Toc501385941)

[**Глава 2. Структура транслятора** 14](#_Toc501385942)

[**2.1 Компоненты транслятора, их назначение и принципы взаимодействия** 14](#_Toc501385943)

[**2.2 Перечень входных параметров транслятора** 15](#_Toc501385944)

[**2.3 Перечень протоколов, формируемых транслятором и их содержимое** 15](#_Toc501385945)

[**Глава 3. Разработка лексического анализатора** 16](#_Toc501385946)

[**3.1 Структура лексического анализатора** 16](#_Toc501385947)

[**3.2 Контроль входных символов** 16](#_Toc501385948)

[**3.3 Удаление избыточных символов** 17](#_Toc501385949)

[**3.4 Перечень ключевых слов, сепараторов, символов операций и соответствующих им лексем, регулярных выражений и конечных автоматов** 17](#_Toc501385950)

[**3.5 Основные структуры данных** 18](#_Toc501385951)

[**3.6 Принцип обработки ошибок** 18](#_Toc501385952)

[**3.7 Структура и перечень сообщений лексического анализатора** 19](#_Toc501385953)

[**3.8 Параметры лексического анализатора и режимы его работы** 19](#_Toc501385954)

[**3.9 Алгоритм лексического анализа** 19](#_Toc501385955)

[**3.10 Контрольный пример** 20](#_Toc501385956)

[**Глава 4. Разработка синтаксического анализатора** 21](#_Toc501385957)

[**4.1 Структура синтаксического анализатора** 21](#_Toc501385958)

[**4.2 Контекстно свободная грамматика, описывающая синтаксис языка** 21](#_Toc501385959)

[**4.3 Построение конечного магазинного автомата** 23](#_Toc501385960)

[**4.4 Основные структуры данных** 24](#_Toc501385961)

[**4.5 Описание алгоритма синтаксического разбора** 24](#_Toc501385962)

[**4.6 Структура и перечень сообщений синтаксического анализатора** 24](#_Toc501385963)

[**4.7 Параметры синтаксического анализатора и режимы его работы** 25](#_Toc501385964)

[**4.8 Принцип обработки ошибок** 25](#_Toc501385965)

[**4.9 Контрольный пример** 25](#_Toc501385966)

[**Глава 5. Разработка семантического анализатора** 26](#_Toc501385967)

[**5.1 Структура семантического анализатора** 26](#_Toc501385968)

[**5.2 Функции семантического анализатора** 26](#_Toc501385969)

[**5.3 Структура и перечень сообщений семантического анализатора** 26](#_Toc501385970)

[**5.4 Принцип обработки ошибок** 27](#_Toc501385971)

[**5.5 Контрольный пример** 27](#_Toc501385972)

[**Глава 6. Преобразование выражений** 27](#_Toc501385973)

[**6.1 Выражения, допускаемые языком** 28](#_Toc501385974)

[**6.2 Польская запись** 28](#_Toc501385975)

[**6.3 Программная реализация обработки выражений** 29](#_Toc501385976)

[**6.4 Контрольный пример** 29](#_Toc501385977)

[**Глава 7. Генерация кода** 30](#_Toc501385978)

[**7.1 Структура генератора кода** 30](#_Toc501385979)

[**7.2 Представление типов данных в оперативной памяти** 30](#_Toc501385980)

[**7.3 Алгоритм работы генератора кода** 30](#_Toc501385981)

[**Глава 8. Тестирование транслятора** 33](#_Toc501385982)

[**8.1 Тестирование фазы проверки на допустимость символов** 33](#_Toc501385983)

[**8.2** **Тестирование лексического анализатора** 33](#_Toc501385984)

[**8.3 Тестирование синтаксического анализатора** 34](#_Toc501385985)

[**8.4 Тестирование семантического анализатора** 34](#_Toc501385986)

[**Приложения** 35](#_Toc501385987)

[**Контрольный пример** 35](#_Toc501385988)

[**Приложение А** 36](#_Toc501385989)

[**Приложение В** 43](#_Toc501385990)

[**Приложение Г** 45](#_Toc501385991)

[**Приложение Е** 50](#_Toc501385992)

[**Литература** 52](#_Toc501385993)

# **Введение**

Целью курсового проекта поставлена задача разработки компилятора для моего языка программирования SVY-2020. Этот язык программирования предназначен для выполнения простейших операций и арифметических действий над числами.

Компилятор SVY-2020 – это программа, задачей которого является перевод программы, написанной на языке программирования SVY-2020 в программу на язык ассемблера.

Транслятор SVY-2020 состоит из следующих частей:

– лексический и семантический анализаторы;

– синтаксический анализатор;

– генератор исходного кода на языке ассемблера.

Исходя из цели курсового проекта, были определены следующие задачи:

– разбработка спецификации языка программирования;

– разбратка структуры транслятора;

– разработка лексического и семантического анализаторов;

– разработка синтаксического анализатора;

– преобразование выражений;

– генерация кода на язык ассемблера;

– тестирование транслятора.

Решения каждой из поставленных задач буду приведены в соответствующих главах курсового проекта.

# **Глава 1. Спецификация языка программирования**

## **Характеристика языка программирования**

Язык программирования SVY-2020 классифицируется как процедурный, универсальный, строготипизированный, компилируемый и не объектно-ориентированный язык.

* 1. **Алфавит языка**

Алфавит языка SVY-2020 основан на кодировке Windows-1251, представленной на рисунке 1.1.



Рисунок 1.1 – Алфавит входных символов

На этапе выполнения могут использоваться символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, спецсимволы, а также непечатные символы пробела, табуляции и перевода строки. Русские символы разрешены только в строковых литералах.

## **Символы сепараторы**

Символы, которые являются сепараторами представлены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |  |
| --- | --- | --- |
| Сепаратор | Название | Область применения |
| ‘ ‘ | Пробел | Допускается везде, кроме идентификаторов и ключевых слов |
| ; | Точка с запятой | Разделение конструкций |
| {…} | Фигурные скобки | Заключение программного блока |
| (…) | Круглые скобки | Приоритет операций, параметры функции |
| ‘…’ | Одинарные кавычки | Допускается везде, кроме идентификаторов и ключевых слов |
| = | Знак «равно» | Присваивание значения |
| , | Запятая | Разделение параметров |
| +  -  \* | Знаки «плюс», «минус», «умножение» | Выражения |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования SVY-2020 используется кодировка Windows-1251.

## **Типы данных**

В языке SVY-2020 реализованы два типа данных: целочисленный и строковый. Описание типов данных, предусмотренных в данным языке представлено в таблице 1.2.

Таблица 1.2 – Типы данных языка SVY-2020

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| Целочисленный тип данных  int | Фундаментальный тип данных. Используется для работы с целочисленными положительными значениями. В памяти занимает 4 байта.  Максимальное значение: 32768.  Минимальное значение: 0.  Инициализация по умолчанию: значение 0. |
| Строковый тип данных string | Фундаментальный тип данных. Используется для работы с символами, каждый символ в памяти занимает 1 байт.  Максимальное количество символов: 255.  Инициализация по умолчанию: длина 0, символ конца строки “\0”. |

## **Преобразование типов данных**

Преобразование типов данных не поддерживается, т.е. язык является строготипизированным.

## **Идентификаторы**

В имени идентификатора допускаются только символы латинского алфавита нижнего регистра. Максимальная длина имени идентификатора - 5 символов. Максимальная длина имени идентификатора функции - 10 символов. При вводе идентификатора длиной более разрешенного количества символов, он будет усекаться. Имя идентификатора не может совпадать с ключевыми словами и не может иметь имя, как функция, уже содержащаяся в стандартной библиотеке.

## **Литералы**

С помощью литералов осуществляется инициализация переменных. В языке существует два типа литералов. Краткое описание литералов языка SVY-2020 представлено в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |  |  |
| --- | --- | --- | --- |
| Тип литерала | Регулярное выражение | Описание | Пример |
| Целочисленный литерал | [1-9]+[0-9]\* | Целочисленные неотрицательные литералы, по умолчанию инициализируются 0. Литералы могут быть только rvalue. | let int sum = 15;  15 – целочисленный литерал. |
| Строковые литерал | [a-z|A-Z|А-Я|а-я|0-9|!-/]+ | Символы, заключённые в ‘’ (одинарные кавычки), по умолчанию инициализируются пустой строкой. Литералы могут быть только rvalue. | let string str = ‘TEXT’;  TEXT – строковый литерал. |

## **Область видимости идентификаторов**

Область видимости «сверху вниз» (по принципу С++). В языке SVY-2020 требуется обязательное объявление переменной перед её инициализацией и последующим использованием. Все переменные должны находиться внутри программного блока. Имеется возможность объявления одинаковых переменных в разных блоках, т. к. переменные, объявленные в одной функции, недоступны в другой. Каждая переменная получает постфикс – название функции, в которой она объявлена.

## **Инициализация данных**

При объявлении переменной не допускается инициализация данных. Краткое описание способов инициализации переменных языка SVY-2020 представлено в таблице 1.4.

Таблица 1.4 – Способы инициализации переменных

|  |  |  |
| --- | --- | --- |
| Конструкция | Описание | Пример |
| let <тип данных> <идентификатор>; | Автоматическая инициализация: переменные типа int инициализируются нулём, переменные типа string – пустой строкой. | let int sum;  let string str; |
| <идентификатор> = <значение>; | Присваивание переменной значения. | sum = 15;  str = ‘TEXT’; |

## **Инструкции языка**

Все возможные инструкции языка программирования SVY-2020 представлены в общем виде в таблице 1.5.

Таблица 1.5 – Инструкции языка программирования SVY-2020

|  |  |
| --- | --- |
| Инструкция | Запись на языке SVY-2020 |
| Объявление переменной | let <тип данных> <идентификатор>; |
| Присваивание | <идентификатор>=<значение>/<идентификатор>; |
| Объявление внешней функции | fn <тип данных> <идентификатор> (<тип данных> <идентификатор>, …) |
| Блок инструкций | main  {  …  } |
| Возврат из подпрограммы | ret <идентификатор> / <литерал>; |
| Условная инструкция | if (условие) [/программный блок если условие верно/]  else [/программный блок если условие ложно/] |
| Вывод данных | output (<идентификатор> / <литерал>); |
| Оператор цикла | cicle(условие) [программный блок] |

* 1. **Операции языка**

Язык программирования SVY-2020 может выполнять арифметические операции, представленные в таблице 1.6.

Таблица 1.6 – Операции языка программирования SVY-2020

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | Примечание | Типы данных | Пример |
| = | Присваивание | (int, int)  (string, string) | sum = 15;  str = ‘TEXT’; |
| > | Знак «больше» для условной инструкции | (int, int) | if (sum > diff) {…}  else {…} |
| < | Знак «меньше» для условной инструкции | (int, int) | if (sum < diff) […]  else […] |
| ( | Приоритет операций | - | sum = (a + b) \* c; |
| ) |
| + | Суммирование | (int, int) | sum = a + b; |
| - | Разность | (int, int) | diff = a – b; |
| \* | Умножение | (int, int) | pr = a \* b; |

## **Выражения и их вычисления**

Круглые скобки в выражении используются для изменения приоритета операций. Также не допускается запись двух подряд идущих арифметических операций. Выражение может содержать вызов функции, если эта функция уже содержится в стандартной библиотеке.

## **Программные конструкции языка**

Ключевые программные конструкции языка программирования SVY-2020 представлены в таблице 1.7.

Таблица 1.7 – Программные конструкции языка SVY-2020

|  |  |
| --- | --- |
| Конструкция | Запись на языке SVY-2020 |
| Главная функция (точка входа) | main  {  …  ret <идентификатор> / <литерал>;  } |
| Функция | fn <тип> <идентификатор> (<тип> <идентификатор>, …)  {  …  ret <идентификатор> / <литерал>;  } |

## **Область видимости**

В языке SVY-2020 все переменные являются локальными. Они обязаны находится внутри программного блока функций (по принципу С++). Объявление глобальных переменных не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

## **Семантические проверки**

Таблица с перечнем семантических проверок, предусмотренных языком, приведена в таблице 1.8.

Таблица 1.8 – Семантические проверки

|  |  |
| --- | --- |
| Номер | Правило |
| 1 | Идентификаторы не должны повторяться |
| 2 | Тип возвращаемого значения должен совпадать с типом функции при её объявлении |
| 3 | Тип данных передаваемых значений в функцию должен совпадать с типом параметров при её объявлении |
| 4 | В функцию должны быть переданы параметры |
| 5 | Тип данных результата выражения должен совпадать с типом данных идентификатора, которому оно присваивается |

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

Все переменные размещаются в стеке.

## **Стандартная библиотека и её состав**

Стандартная библиотека SVY-2020 написана на языке программирования C++. Функции стандартной библиотеки с описанием представлены в таблице 1.9.

Таблица 1.9 – Состав стандартной библиотеки

|  |  |  |
| --- | --- | --- |
| Функция(C++) | Возвращаемое значение | Описание |
| string strcat(string, string); | string | Функция объединяет две строки |
| int transform(string str); | int | Функция вычисляет длину строки str |
| int output(string str) | 0 | Функция вывода на консоль строкового идентификатора/литерала |

## **Ввод и вывод данных**

В языке SVY-2020 не реализованы средства ввода данных.

Для вывода данных в стандартный поток вывода предусмотрены функция output, которая входит в состав стандартной библиотеки и описаны в таблице 1.9.

## **Точка входа**

В языке SVY-2020 каждая программа должна содержать главную функцию main, т. е. точку входа, с которой начнется последовательное выполнение программы.

## **Препроцессор**

Препроцессор в языке программирования SVY-2020 не предусмотрен.

## **Соглашения о вызовах**

В языке вызов функций происходит по соглашению о вызовах stdcall. Особенности stdcall:

– все параметры функции передаются через стек;

– память высвобождает вызываемый код;

– занесение в стек параметров идёт справа налево.

## **Объектный код**

SVY-2020 транслируется в язык ассемблера.

## **Классификация сообщений транслятора**

В случае возникновения ошибки в коде программы на языке SVY-2020 и выявления её транслятором в текущий файл протокола выводится сообщение. Их классификация сообщений приведена в таблице 1.10.

Таблица 1.10. Классификация сообщений транслятора

|  |  |
| --- | --- |
| Интервал | Описание ошибок |
| 0-99 | Системные ошибки |
| 100-109 | Ошибки параметров |
| 110-119 | Ошибки открытия и чтения файлов |
| 120-599 | Ошибки лексического анализа |
| 600-699 | Ошибки синтаксического анализа |
| 700-900 | Ошибки семантического анализа |

## **Контрольный пример**

Контрольный пример представлен в главе Приложения.