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 SVA-2022»

Выполнил студент Смолик Валерий Александрович

(Ф.И.О.)

Руководитель проекта ст.пр. Наркевич Аделина Сергеевна

(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 ст.пр. Наркевич Аделина Сергеевна

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

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

Нормоконтролер ст.пр. Наркевич Аделина Сергеевна

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

Курсовой проект защищен с оценкой

Минск 2020

Оглавление

[**Введение** 2](#_Toc119407913)

[**Глава 1. Спецификация языка программирования** 3](#_Toc119407914)

[**1.1** **Характеристика языка программирования** 3](#_Toc119407915)

[**1.2** **Определение алфавита языка программирования** 3](#_Toc119407916)

[**1.3 Применяемые сепараторы** 3](#_Toc119407917)

[**1.4** **Применяемые кодировки** 4](#_Toc119407918)

[**1.5 Типы данных** 5](#_Toc119407919)

[**1.6 Преобразование типов данных** 5](#_Toc119407920)

[**1.7 Идентификаторы** 5](#_Toc119407921)

[**1.8 Литералы** 6](#_Toc119407922)

[**1.9 Объявление данных** 6](#_Toc119407923)

[**1.10 Инициализация данных** 6](#_Toc119407924)

[**1.11 Инструкции языка** 7](#_Toc119407925)

[**1.12 Операции языка** 7](#_Toc119407926)

[**1.13** **Выражения и их вычисления** 7](#_Toc119407927)

[**1.14** **Конструкции языка** 8](#_Toc119407928)

[**1.15** **Область видимости идентификаторов** 8](#_Toc119407929)

[**1.16** **Семантические проверки** 8](#_Toc119407930)

[**1.17** **Распределение оперативной памяти на этапе выполнения** 8](#_Toc119407931)

[**1.18** **Стандартная библиотека и ее состав** 8](#_Toc119407932)

[**1.19 Вывод и ввод данных** 9](#_Toc119407933)

[**1.20 Точка входа** 9](#_Toc119407934)

[**1.21 Препроцессор** 9](#_Toc119407935)

[**1.22 Соглашения о вызове** 9](#_Toc119407936)

[**1.23 Объектный код** 9](#_Toc119407937)

[**1.24** **Классификация сообщений транслятора** 9](#_Toc119407938)

[**1.25 Контрольный пример** 9](#_Toc119407939)

# **Введение**

Транслятор – это комплекс отдельных программ, позволяющих преобразовывать исходный код на одном языке программирования в исходный код на другом языке программирования.

Классический транслятор состоит из следующих частей:

* лексический анализатор;
* синтаксический анализатор;
* семантический анализатор;
* генератор кода, или интерпретатор.

Все части транслятора, взаимодействуя между собой, обрабатывают входной текст и строят для него эквивалентный текст на понятном компьютеру языке программирования.

**Глава 1. Спецификация языка программирования**

* 1. **Характеристика языка программирования**

Язык программирования SVA-2022 является языком программирования высокого уровня. Он является компилируемым. В языке отсутствует преобразование типов. В языке поддерживается 2 типа данных: целочисленный (number) и строковый (symbols). В стандартной библиотеке имеются функции для работы с целочисленным и символьным типами данных: генерация случайных чисел, вычисление длины строки.(парадигма)

* 1. **Определение алфавита языка программирования**

Символы, используемые на этапе выполнения: [a…z], [A…Z], [0…9], [a…я]символы пробела, табуляции и перевода строки, спецсимволы: [] () , ; : + - / \* % > < !. множества

**1.3 Применяемые сепараторы**

Символы сепараторы служат в качестве разделителей цепочек языка во время обработки исходного текста программы с целью разделения на токены. Они представлены в таблице 1.1.

Таблица 1.1 – Символы-сепараторы

|  |  |
| --- | --- |
| Символ(ы) | Назначение |
| Пробел | Разделитель цепочек. Допускается везде, кроме имен идентификаторов и ключевых слов |
| **[** … **]** | Блок функции или цикла |
| **(** … **)** | Блок параметров функции |
| **,** | Разделитель параметров функций |
| **+ - \* /** | Арифметические операции |
| **> < ! } {** | Логические операторы (Операции сравнения: больше, меньше, не равно, больше или равно, меньше или равно) |
| **;** | Разделитель программных конструкций |
| **=** | Оператор присваивания |

* 1. **Применяемые кодировки**

Для написания программ на языке SVA-2022 используется кодировка Windows – 1251, представленная на рис.1.1.

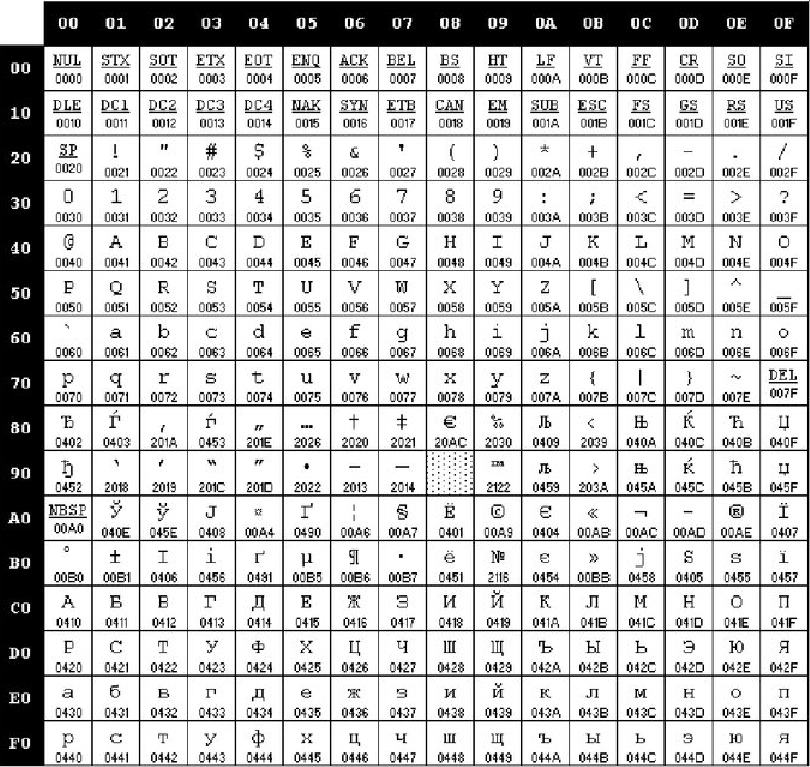


Рисунок 1.1 Алфавит вводных символов

**1.5** **Типы данных**

В языке SVA-2022 реализованы 2 фундаментальных типа данных: целочисленный, строковый. Описание типов приведено в таблице 1.2.

Таблица 1.2 – Типы данных языка SVA-2022

|  |  |
| --- | --- |
| Типы данных | Характеристика |
| Целочисленный тип данных **numb** | Является целочисленным типом данных. Этот тип данных занимает 4 байта. Предназначен для арифметических операций над числами. Инициализация по умолчанию: 0.  Поддерживаемые операции:  + (бинарный) – оператор сложения;  - (бинарный) – оператор вычитания;  \* (бинарный) – оператор умножения;  / (бинарный) – оператор деления;  = (бинарный) – оператор присваивания  %(бинарный) – оператор деления по модулю  добавить |
| строковый тип данных **stroke** | Фундаментальный тип данных. Используется для работы с символами, каждый из которых занимает 1 байт. Максимальное количество символов – 255.  Инициализация по умолчанию: строка нулевой длины “”.  Операции над данными строкового типа:  = (бинарный) – присваивание значения |

**1.6 Преобразование типов данных**

Преобразование типов данных в языке SVA-2022 не поддерживается, так как язык SVA-2022 является типизированным.

**1.7 Идентификаторы**

Общее количество идентификаторов ограниченно максимальным размером таблицы идентификаторов (4096). Идентификаторы могут содержать символы как нижнего регистра, так и верхнего. Максимальная длина идентификатора равна 10 символам. Идентификаторы, объявленные внутри функционального блока, получают префикс, идентичный имени функции, внутри которой они объявлены. Данные правила действуют для всех идентификаторов. Зарезервированные идентификаторы не предусмотрены. Идентификаторы не должны совпадать с ключевыми словами. Типы идентификаторов: имя переменной, имя функции, параметр функции.

Правило составления идентификатора:

<идентификатор> ::= <прописная буква латинского алфавита>

**1.8 Литералы**

С помощью литералов осуществляется инициализация переменных. Все литералы являются rvalue. Имеются литералы: целочисленные литералы десятичного представления, а так же строковые литералы. Подробное описание литералов языка SVA-2022 представлены в таблице  1.3.

Таблица 1.3 – Литералы

|  |  |
| --- | --- |
| Литералы | Пояснение |
| Целочисленные литералы в десятичном представлении | Последовательность цифр 0…9 с предшествующим знаком минус или без него (знак минус не отделяется пробелом)добавить представ |
| Строковые литералы | Набор символов алфавита языка, заключенных в двойные кавычки |

Ограничения на целочисленные литералы: не могут начинаться с 0, если их значение не 0; если литерал отрицательный, после знака “-” не может идти 0. Подумать над -

**1.9** **Объявление данных**

Для объявления переменной используется ключевое слово **new,** после которого указывается тип данных и имя идентификатора.

Пример объявления числового типа данных с инициализации:

**new number** x = 14;

Пример объявления строкового типа данных с инициализацией:

**new symbols** str = 'привет мир';

Для объявления функций используется ключевое слово **func**, перед которым указывается тип функции. Далее обязателен список параметров и тело функции.

**1.10 Инициализация данных**

При объявлении переменной допускается инициализация данных. При этом переменной будет присвоено значение литерала или идентификатора, стоящего справа от знака равенства. Объектами-инициализаторами могут быть только идентификаторы и литералы. При объявлении переменные инициализируются значением по умолчанию. Для **number** значение 0, для **text** строка нулевой длины (“”).

**1.11** **Инструкции языка**

Инструкции языка SVA-2022 представлены в таблице 1.4.

Таблица 1.4 – Инструкции языка

|  |  |
| --- | --- |
| Инструкция | Реализация |
| Объявление переменной | **New** <тип данных>< идентификатор >; |
| Возврат значения из функции | **return** <идентификатор> | <литерал>; |
| Вывод данных | **print** <идентификатор> | <литерал>; |
| Вызов функции | <идентификатор функции>(<список параметров>); |
| Присваивание | <идентификатор> = <выражение>;  Выражением может быть идентификатор, литерал, или вызов функции соответствующего типа. Для целочисленного типа выражение может быть дополнено арифметическими операциями. |

**1.12 Операции языка**

Операции языка SVA-2022 и их приоритет представлен в таблице 1.5.

Таблица 1.5 – Операции языка SVA-2022

|  |  |
| --- | --- |
| Тип оператора | Оператор |
| Арифметические | **+ -** сложение (приоритет 3)  - **-** разность (приоритет 3)  **\* -** умножение (приоритет 2)  **/ -** деление (приоритет 2)  % - деление по модулю (приоритет 2) |
| Логические | **>** - больше (приоритет 4)  **<** - меньше (приоритет 4)  **!** – не равно (приоритет 5)  **}** – больше или равно (приоритет 4)  **{** - меньше или равно (приоритет 4) |

Для повышения приоритета выполнения операций используются круглые скобки “( )”.

* 1. **Выражения и их вычисления**

Вычисление выражений – одна из важнейших задач языков программирования. Всякое выражение составляется согласно следующим правилам:

1. Допускается использовать скобки для смены приоритета операций;

2. Выражение записывается в строку без переносов;

3. Использование двух подряд идущих операторов не допускается;

4. Допускается использовать в выражении вызов функции, вычисляющей и возвращающей целочисленное значение.

* 1. **Конструкции языка**

Программа на языке SVA-2022 оформляется в виде функций пользователя и главной функции. При составлении функций рекомендуется выделять блоки и фрагменты и применять отступы для лучшей читаемости кода.

Программные конструкции языка SVA-2022 представлены в таблице 1.6.

Таблица 1.6 – Конструкции языка SVA-2022

|  |  |
| --- | --- |
| Конструкция | Реализация |
| Главная функция | **main**  **[**  …  **]** |
| Внешняя функция | <тип данных> **func** <идентификатор> (<тип> <идентификатор>, ...)  **[**  …  **return** <идентификатор/литерал>;  **]** |

* 1. **Область видимости идентификаторов**

Область видимости: сверху вниз. Переменные, объявленные в одной функции не доступны в другой. Все операции и объявления происходят внутри какого-либо блока или тела функции. Каждая переменная или параметр функции получают префикс – название функции, в которой они находятся.

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

* 1. **Семантические проверки**
  2. **Распределение оперативной памяти на этапе выполнения**
  3. **Стандартная библиотека и ее состав**

В языке SVA-2020 присутствует стандартная библиотека, которая подключается автоматически на этапе трансляции исходного кода в язык ассемблера.

Содержимое стандартной библиотеки представлено в таблице 1.8.

Таблица 1.8 – Стандартная библиотека языка SVA-2022

|  |  |
| --- | --- |
| Функция | Описание |
| **number** randnum() | Целочисленная функция, возвращает псевдослучайное число |
| **number** len(**symbols** x) | Целочисленная функция, возвращает размер строки |

**1.19 Вывод и ввод данных**

Вывод данных осуществляется с помощью оператора **print**. Допускается использование оператора **print** с литералами и идентификаторами. Допускается перевод строки при использовании функции **printline**.ред

**1.20 Точка входа**

В языке SVA-2022 каждая программа должна содержать главную функцию (точку входа) **main**, с первой инструкции которой начнётся последовательное выполнение команд программы.

Должна иметься только одна точка входа main.

**1.21 Препроцессор**

Команды препроцессора в языке SVA-2022 отсутствуют.

**1.22 Соглашения о вызове**

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.

**1.23** **Объектный код**

* 1. **Классификация сообщений транслятора**

**1.25 Контрольный пример**