МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информацион-

ных технологий

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора WSA-2020»

Выполнил студент Валько Сергей Александрович

(Ф.И.О. студента)

Руководитель проекта ст. преп. Наркевич Аделина Сергеевна

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

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

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

Консультанты Фамилия Имя Отчество

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

Курсовой проект защищен с оценкой

Минск 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующая кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Языки программирования"

Специальность: ПОИТ Группа: 7 Студент: Валько Сергей Александрович

(фамилия, имя, отчество)

**1. Тема проекта** Разработка компилятора WSA-2020

утверждена приказом по университету от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. №

**2. Срок сдачи студентом законченного проекта:** \_\_\_ декабря 2020 г.

**3. Исходные данные к проекту:**

Разработка программы осуществляется на языке C++ (стандартизации International Standard ISO/IEC 14882:2017 Programming Language C++ 17) в среде разработки Visual Studio 2019 (v142). Операционная система под которой происходит разработка Windows 10 home (64-bit). Типы данных: short и string. Функции стандартной библиотеки: short len(string) – длина строки, short random(short, short) – генерация случайного числа. Арифметические операции: +, -, \*, /, %. Оператор вывода в стандартный поток. 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)

Приложения

Литература

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

1) Граф

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

|  |  |
| --- | --- |
| Раздел | Консультант |
| Разработка синтаксического и семантического анализатора. | Наркевич А. С. |
| Генерация кода. Разработка тестовых примеров. | Наркевич А. С. |
| Оформление пояснительной записки к курсовому проект. | Наркевич А. С. |
|  |  |

**7. Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Спецификация специализированного языка WSA-2020 |  |  |
| 2 | Разработка лексического анализатора |  |  |
| 3 | Разработка синтаксического анализатора |  |  |
| 4 | Разработка семантического анализатора |  |  |
| 5 | Генерация кода |  |  |
| 6 | Тестирование компилятора |  |  |
| 7 | Оформление пояснительной записки к курсовому проект |  |  |
| 8 | Сдача проекта | 17.12.2020 |  |

**8. Дата выдачи задания 15.09.2020**

Руководитель Наркевич А. С.

(подпись)

Задание принял к исполнению Валько С. А.

(дата и подпись студента)

**Глава 1.Спецификация языка программирования**

* 1. **Характеристика языка программирования**

Язык WSA-2020 – это процедурный, не строго типизированный со статической типизацией и обязательной декларацией, компилируемый язык высокого уровня.

* 1. **Определение алфавита языка программирования**

Алфавит языка WSA-2020 содержит следующие множества:

* прописные и строчные латинские буквы: [a-z, A-Z];
* **прописные и строчные буквы кириллицы: [а-я, А-Я];**
* цифры: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
* специальные символы: {\* + - ? ‘ ,: ; $ # = % ( ) { } < > !};
* пробельный символ;
* символ перехода на новую строку.
  1. **Символы сепараторы**

Символы–сепараторы представлены в таблице 1.1.

Таблица 1.1 Символы-сепараторы

|  |  |
| --- | --- |
| Символ-сепаратор | Характеристика |
| ; | Разделяет конструкции языка. |
| Пробел | Допускается везде кроме идентификаторов и ключевых слов. |
| () | Определяют приоритет операций. Содержат параметры функции или процедуры. |
| {} | Программный блок. |

* 1. **Применяемые кодировки**

В языке WSA-2020 используется кодировка Windows-1251 (CP1251).

Таблица кодировки изображена на рисунке 1.1.

Рисунок 1.1 Таблица кодировки Windows-1251(CP1251).



**1.5 Типы данных**

Язык WSA-1251 поддерживает целочисленный, строковый, символьный и логический типы данных. Описание типов данных, предусмотренных в данном языке, представлено в таблице 1.2.

Таблица 1.2 Типы данных языка программирования WSA-2020

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип данных | Размер | Диапазон значений | Значение по умолчанию | Описание |
| short | 2 байта | {-32 768…32 767} | 0 | Фундаментальный тип данных, предназначенный для хранения целых чисел. |
| string | 1 символ -  1 байт | - | "" | Фундаментальный тип данных, предназначенный для представления символьных строк. Максимальная длина строки: 256 символов. |
| bool | 1 байт | true и false | false | Фундаментальный тип данных, содержащий значения: true(истина) и false(ложь). |
| char | 1байт | {-128 … 127} | '' | Фундаментальный тип данных, предназначенный для хранения символьной информации |

* 1. **Преобразование типов**

В языке WSA-2020 отсутствует преобразование типов.

**1.7 Идентификаторы**

В имени идентификатора используются символы латинского алфавита нижнего регистра и цифры (цифра не должна находиться на первой позиции). Максимальная длина идентификатора 8 символов. В случае, если эта длина превышена, имя идентификатора урезается. Пример, верно, записанного идентификатора: value123. Пример, неверно записанного идентификатора: 12val.

**1.8 Литералы**

В языке WSA-2020 присутствует 4 типа литералов: логические, целочисленные (двоичное и восьмеричное представление), строковые и символьные. Краткое описание литералов представлено в таблице 1.3.

Таблица 1.3 Описание литералов

|  |  |
| --- | --- |
| Литерал | Описание |
| Целочисленный | Восьмеричное представление: 0oX или 1oX, X – число в восьмеричной системе счисления .  Двоичное представление: 0bX или 1bX, X - число в двоичной системе счисления.  (0 – положительное число, 1 – отрицательное число)  Диапазон значений: от -32 768 до 32 767. |
| Строковый | Символы, помещённые в "" (код в Windows-1251: 34).  Максимальное количество символов: 255. |
| Логические | true (значение - 1)  false (значение - 0) |
| Символьные | Символ, помещённый в ' ' (код в Windows-1251: 39). |

Пример целочисленного литерала: 0b111001, 0o1234.

Пример строкового литерала: ' dggfgfdhd '.

**1.9 Объявление данных**

Конструкция объявления переменной: var <тип данных> <идентификатор>;

**1.10 Инициализация данных**

Язык WSA-2020 не поддерживает инициализацию данных.

Язык предусматривает инициализацию по умолчанию (сведения об инициализации по умолчанию для конкретных типов данных предоставлены в таблице 1.2).

**1.11 Инструкции языка**

Инструкции языка программирования WSA-2020 представлены в таблице 1.4.

Таблица 1.4 Инструкции языка программирования WSA-2020

|  |  |
| --- | --- |
| Инструкция | Синтаксис |
| Главная функция | main  {  <инструкции языка>; } |
| Объявление переменной | var <тип данных> <идентификатор>; |
| Объявление функции | func <тип данных> <идентификатор> (<параметр>, …)  {  <инструкции языка>; } |
| Параметр функции(процедуры) | <тип данных> <идентификатор> |
| Объявление процедуры | proc<идентификатор> (<параметр>, …)  {  <инструкции языка>; } |
| Операция присваивания | <идентификатор>=<идентификатор>;  <идентификатор>=<литерал>;  <идентификатор>=<выражение>; |
| Вызов функции (процедуры) | $<идентификатор>(<идентификатор >|<литерал>, …); |
| Возврат значения | ret <идентификатор>|<литерал>|<выражение>; |
| Оператор цикла | while(условие)  {  <инструкции языка>; } |
| Вывод данных | write <идентификатор>|<литерал>|<выражение>; |
| Подключение внешней функции | extr <тип данных>  <идентификатор>(<параметр>,…); |

**1.12 Операции языка**

Операции языка программирования WSA-2020 представлены в таблице 1.5.

Таблица 1.5 Операции языка программирования WSA-2020

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Тип оператора | Тип данных | Описание |
| + | бинарный | short | Сложение двух целых чисел |
| + | бинарный | string | Конкатенация строк |
| - | унарный | short | Установка знака числа в противоположный |
| - | бинарный | short | Вычитание целых чисел |
| \* | бинарный | short | Умножение целых чисел |
| / | бинарный | short | Деление нацело |
| % | бинарный | short | Деление по модулю |
| < | бинарный | short | Проверяет меньше ли правое число левого |
| > | бинарный | short | Проверяет больше ли правое число левого |
| < | бинарный | string | Проверяет меньше ли правая строка левой по длине |
| > | бинарный | string | Проверяет больше ли правая строка левой по длине |
| == | бинарный | short | Проверяет числа на равенство |
| == | бинарный | string | Проверяет строки на равенство по длине |
| == | бинарный | bool | Проверяет логические значения на равенство |
| != | бинарный | bool, string, short | Производит действия обратные оператору ‘==’ |

Приоритет операций (с верху в низ):

1.()

2.== \* / != < > %

3. + -

**1.13 Выражения и их вычисление**

Выражение – выполнение операций над операндами. Операнды могут быть представлены переменными, литералами или вызовом функций. Признаком начала выражения является оператор присваивания, а конца – точка с запятой. Результатом выражения является значение определённого типа. Операнды, над которыми выполняется операция, должны иметь одинаковые типы.

**1.14 Конструкции языка**

Программные конструкции языка WSA-2020 представлены в таблице 1.6.

Таблица 1.6 Конструкции языка программирования WSA-2020

|  |  |
| --- | --- |
| Конструкция | Синтаксис |
| Главная процедура (точка входа) | main  {  <инструкции языка>; } |
| Функция | <тип данных>  func<идентификатор>(<параметры>)  { <инструкции языка>;  ret <идентификатор>|<литерал>;  } |
| Процедура | proc<идентификатор>(<параметры>)  {  <инструкции языка>; } |

**1.15 Область видимости идентификаторов**

Переменные могут определяться внутри программного блока. Переменные являются локальными. Функции определяются в глобальной области видимости. Объявление пользовательских областей видимости не предусмотрено.

**1.16 Семантические проверки**

В языке WSA-2020 предусмотрены следующие семантические правила:

* переменная или константа должна быть объявлена перед использованием;
* объявленная переменная должно хотя бы раз использоваться в выполняемом коде;
* исходный текст должен обязательно содержать точку входа main;
* результат функции должен быть определён при любом ходе её выполнения;
* оператор цикла должен предусматривать возможность обязательного окончания работы;
* не допускаются идентификаторы, совпадающие с ключевыми словами;
* функция или процедура перед выполнением требует обязательного объявления;
* не разрешается повторное объявление функции или процедуры.

**1.17 Распределение оперативной памяти на этапе выполнения**

Транслированный код использует следующие области памяти: сегмент констант и сегмент данных. Сегмент констант содержит литералы, а сегмент данных переменные, параметры функций и возвращаемые значения функции. Локальная область видимости в исходном коде определяется за счет использования правил именования идентификаторов и регулируется их префиксами.

**1.18 Стандартная библиотека и её состав**

Стандартная библиотека языка WSA-2020 представлена в таблице 1.7.

Таблица 1.7 Стандартная библиотека языка программирования WSA-2020

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Тип возвращаемого значения | Параметры | Описание |
| len | short | string | Возвращает длину полученной строки |
| random | short | short, short | Возвращает случайное число в диапазоне, заданном параметрами функции. |
| writeOct | bool | short | Выводит число в восьмеричном формате. |

Синтаксис подключения функции стандартной библиотеки:

extr <тип данных><идентификатор>(<параметры>);

**1.19 Ввод и вывод данных**

В языке WSA-2020 предусмотрен вывод данных в консоль. Вывод осуществляется с помощью ключевого слова ‘write’, его синтаксис:

write <идентификатор>|<вызов функции>|<литерал>;

Оператор ввода в языке не предусмотрен.

**1.20 Точка входа**

Точка входа в языке WSA-2020 представлена главной процедурой, которая обозначается ключевым словом ‘main’.

**1.21 Препроцессор**

В языке WSA-2020 препроцессор не предусмотрен.

**1.22 Соглашения о вызовах**

Используется соглашение о вызовах stdcall:

* параметры передаются через стек;
* параметры следуют справа налево;
* стек очищается вызываемым кодом.

**1.23 Объектный код**

Целевой язык трансляции - язык ассемблера для процессоров Intel семейства IA-32.

**1.24 Классификация сообщений транслятора**

Классификации сообщений транслятора содержится в таблице 1.8

Таблица 1.8 Классификация сообщений транслятора

|  |  |
| --- | --- |
| Кодовое значение ошибки | Описание |
| 0-200 | Системные ошибки |
| 200 – 300 | Ошибки на этапе лексического анализа |
| 300 - 400 | Ошибки на этапе синтаксического анализа |
| 600 – 700 | Ошибки на этапесемантического анализа |
| 400-500, 700 - 1000 | Зарезервированные коды ошибок |

**1.25 Контрольный пример**

Контрольный пример кода на языке WSA-2020 находится в приложении А.

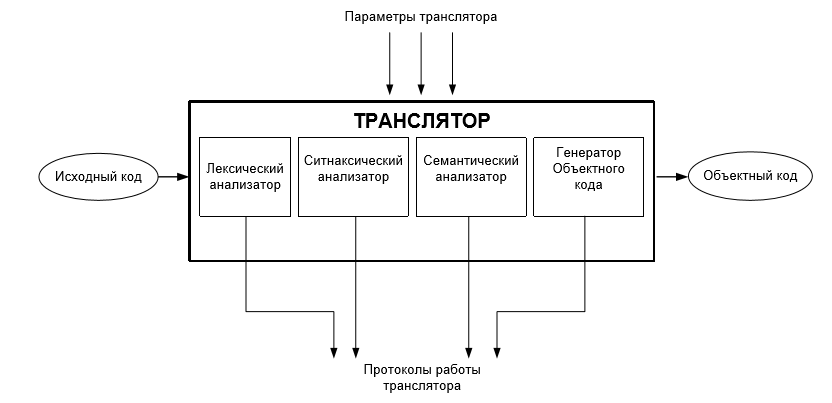
**Глава 2. Структура транслятора**

Транслятор преобразует исходный текст программы в текст целевого языка. Получив на вход исходный текст, транслятор проверяет его принадлежность заданному языку и определяет набор грамматических правил языка. Процесс трансляции состоит из фаз: лексический анализ, синтаксический анализ, семантический анализ и генерация кода. На всех фазах трансляции применяется таблица идентификаторов, которая пополняется дополнительной информацией в ходе трансляции.

**2.1. Компоненты транслятора их назначение и принципы взаимодействия**

Транслятор преобразует программу, написанную на языке WSA-2020, в язык ассемблера. Его составные части: лексический анализатор, синтаксический анализатор, семантический анализатор и генератор кода на язык ассемблера. Принцип их взаимодействия представлен на рисунке 2.1.

Рисунок 2.1 Принцип взаимодействия составных частей транслятора

****

Лексический анализатор принимает на вход исходный код программы, а по окончании его работы на выходе получаем: таблицу лексем, таблицу идентификаторов и протокол работы. Задача лексического анализатора проверить на правильность входные символы и лексические единицы (лексемы).

Синтаксический анализатор принимает на вход таблицу лексем и идентификаторов, а на выходе выдаёт дерево разбора. Задача синтаксического анализатора проверить правильность цепочек языка.

Семантический анализатор получает таблицу лексем и идентификаторов и проверяет входные данные на соблюдение смысловых нагрузок.

Генератор кода принимает результаты работы предыдущих этапов работы транслятора и на их основе генерирует код на языке ассемблера.

**2.2 Перечень входных параметров транслятора**

Входные параметры транслятора описаны в таблице 2.1.

Таблица 2.1. Перечень входных параметров транслятора

|  |  |  |
| --- | --- | --- |
| Параметр | Характеристика | Значение по умолчанию |
| -in:<путь к файлу> | Описывает путь к исходному файлу на языке WSA-2020. | Не предусмотрено. |
| -log:<путь к файлу> | Описывает путь к log файлу для вывода протоколов работы транслятора. | <имя in-файла>.log |
| -out:<путь к файлу> | Описывает путь к out файлу с результирующим кодом на языке ассемблера. | <имя in-файла>.asm |

**2.3 Протоколы, формируемые транслятором**