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 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 strlen(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)

1.Спецификация языка программирования

1.1 Характеристика языка программирования

* Простой язык программирования решающий базовые задачи.
* Язык высокого уровня, транслируемый на язык ассемблера.
* Является компилируемым, исходный текст анализируется и разбирается целиком.
* Поддерживает структурную парадигму программирования.
* Нестрого типизированный язык с статической типизацией и обязательной декларацией.

1.2 Определение алфавита языка программирования

* Алфавит языка включает латиницу и кириллицу в верхнем и нижнем регистрах, а также цифры от 0 до 9 и специальные символы.
* A B C D E F G H I K L M N O P Q R S T V X Y Z a b c d e f g h I k l m n o p q r s t v x y z
* **А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я**
* 0 1 2 3 4 5 6 7 8 9
* \* + - ? ‘ , ; $ # = % ( ) { } < > !
* Пробельный символ
* Символ перехода на новую строку

1.3 Символы сепараторы

* ; - разделитель конструкций.
* Пробел – допускается везде кроме идентификаторов и ключевых слов.
* () – содержат параметры функции.
* () - приоритет операций.
* {} – программный блок.

1.4 Применяемые кодировки

* Windows-1251.

1.5 Типы данных

* Язык поддерживает целочисленный, строковый и булевский типы данных
* short – знаковый целочисленный тип данных. В памяти под значения данного типа выделяется область размером 2 байта. Диапазон значений: -32 768 и 32 767. По умолчанию принимает значение 0.
* string – строковый тип данных. Представляет, из себя набор символов. В памяти представлена как последовательность символов под каждый из которых выделена ячейка размером 1 байт. Максимальное количество символов в строке 256. Не принимает значений по умолчанию.
* bool – логический тип данных, который может принимать два значения true и false (правда и ложь). В памяти занимает ячейку размером 1 байт, true в памяти соответствует числовое значение 1, а false 0.

1.6 Преобразование типов

* Поддерживается неявное преобразование из типа short в bool и из bool в short. В первом случае значение short преобразовывается в true, если значение больше или равно 1, иначе преобразование осуществляется к false. Во втором случае true преобразовывается к 1, а false к 0.
* Поддерживается явное преобразование short к string. Число преобразовывается в свой символьный аналог.
* Поддерживается явное преобразование от bool к string. Значение true преобразовывается в ‘true’, а false в ‘false’.

1.7 Идентификаторы

* В идентификаторе используются символы, обозначающие буквы английского алфавита в нижнем регистре, а в случае именованных констант в верхнем.
* Максимальная длина идентификатора – 8 символов.
* Примеры правильных идентификаторов: value, ptr, param и т.д.
* Примеры неверных идентификаторов: 12val, \_kop, значение, pop corn и т.д.

1.8 Литералы

* Представляют, из себя числовое значение или строку.
* Диапазон значений целочисленного литерала: от -32 768 до 32 767.
* Целочисленный литерал представляет знаковое или без знаковое число.
* Строковый литерал представлен символьной строкой заключённой в одинарные кавычки.
* Строковый литерал может содержать строку максимум из 256 символов.
* Строковый литерал может содержать символы как кириллицы так и латиницы в верхнем и нижних регистрах, пробелы, а также спецсимволы алфавита языка.
* Примеры: 12243, -2134, ‘авпапвапвап’, ‘dggrwgwfew’.

1.9 Объявление данных

* Объявление переменной начинается с ключевого слова var за которым следует объявление типа и идентификатор. Пример: var short value.
* Переменная может быть объявлена в глобальной области видимости, в области видимости главной функции или в областях видимости пользовательских функций и процедур.
* Объявление пользовательской функции начинается с ключевого слова func, далее следует тип возвращаемых данных, идентификатор и объявление параметров в круглых скобках. Объявление параметров содержит тип данных и идентификатор. После объявления параметров находится тело функции в фигурных скобках. Пример: function string strcat(string s1, string s2){return s2;}.
* Объявление процедуры начинается с ключевого слова proc за которым следует идентификатор и объявление параметров (аналогично объявлению параметров функции). После объявления параметров находится тело процедуры в фигурных скобках. Пример: proc write(string nm).

1.10 Инициализация данных

* = - оператор присваивания, используется для инициализации переменных определённым значением.
* Для успешной инициализации тип значения до оператора присваивания и тип значения после этого оператора должны совпадать, либо быть приводимыми друг к другу.
* По умолчанию инициализируются целочисленные и булевские переменные(0 и false).

1.11 Инструкции языка

* var - объявление переменной. Используется для определения новой переменной, может использоваться в главной области, либо в пользовательских функциях или процедурах. Пример:

var a; var value = 1;

* = - оператор присваивания. Помещает значение rvalue в lvalue. Необходимо чтобы значения были одного типа либо приводимы один к одному. Пример: var short val = abc; a = b; a = b+c.
* write – вывод в консоль. Пример: write value;
* while – цикл. Выполняет последовательность действий пока верно некоторое условие. Шаблон: write(условие){действия}.
* ret – вернуть значение из функции. Пример: ret var string str = “HELLO”;
* Последовательность инструкций начинается оператором присваивания и заканчивается ‘;’ .

1.12 Операции языка

* + - бинарный, производит сложение двух целочисленных значений.
* - - унарный, меняет знак значения на противоположный, применим к типу short.
* - - бинарный, производит вычитание правого значения из левого, применим к типу short.
* / - бинарный, производит деление нацело, применим к типу short.
* % - бинарный, производит деление по модулю, применим к типу short.
* \* - бинарный, производит перемножение значений, применим к типу short.
* < - бинарный, возвращает true если lvalue меньше rvalue (для строк сравнение происходит по длине).
* > - бинарный, возвращает true если lvalue больше rvalue (для строк сравнение происходит по длине).
* == - бинарный, возвращает true если lvalue и rvalue равны (для строк сравнение происходит по длине).
* != - - бинарный, возвращает true если lvalue и rvalue неравны (для строк сравнение происходит по длине).
* Приоритет операций(по нисходящей):

1.< > == != \* / %

2. + -

* Для присваивания наибольшего приоритета операции следует заключить выражение в ().

1.13 Выражения и их вычисление

* Выражение языка начинается с оператора присваивания = и заканчивается ‘ ; ’ .
* Минимальное выражение: lvalue = rvalue;
* Операндами выражения могут быть литералы, переменные, а также вызовы функций или процедур.
* Операнды должны быть одного типа, либо приводимы один к одному.
* Выражение производит вычисление некоторого значения при помощи разрешённых языком операций над операндами.
* Выполнение выражения происходит слева направо.

1.14 Конструкции языка

* func – объявление функции. Используется для определения пользовательских функций. Должно находиться в глобальной области видимости.
* Шаблон объявления функции:

func тип идентификатор(параметры){набор инструкций};

* В набор инструкций обязательно входит возврат результат работы функции.
* При объявлении параметров нужно указать их тип, разделяются запятыми. Максимальное количество параметров: 10.
* proc – объявление процедуры. Используется для определения пользовательских процедур. Должно находиться в глобальной области видимости.
* Шаблон объявления процедуры:

proc идентификатор(параметры){набор инструкций};

* $ - вызов функции или процедуры. Используется для вызова пользовательских/стандартных функций или процедур. Пример:

$strlen(“SWA-2020”);

* Принимаемым параметром может быть функция возвращающая значение определённого типа, но процедура не может передаваться в качестве параметра.
* Основная область выполнения исходного кода представлена конструкцией main, которая по сути является функцией. При успешном выполнении программы возвращается 1, иначе 0.

Пример:

main

{

var short a;

var string b = “fgafga”;

write b;  
}

1.15 Область видимости идентификаторов

* Идентификаторы переменной видны в рамках области видимости, в которой определены.
* Переменные не могут быть объявлены в глобальной области видимости.
* Идентификаторы функций и процедур имеют глобальную область видимости и видны в рамках всего исходного кода.
* Идентификаторы параметров видны в рамках функции (процедуры), в которой объявлены.

1.16 Семантические проверки

* 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

1.18 Стандартная библиотека и её состав

* Функции стандартной библиотеки содержаться в файле подключаемом через директиву препроцессора.
* Функции стандартной библиотеки находятся в файле standart.txt.
* Стандартная библиотека содержит 2 функции: short strlen(string), short random(short, short).
* strlen – возвращает длину передаваемой строки. Шаблон функции: short strlen(string). Тип возвращаемого значения: short. Принимает параметр типа string.
* random – функция возвращает случайное число типа short. Принимает на вход два параметра типа short. Первый задаёт начальную границу диапазона случайных чисел, а второй конечную границу. Шаблон функции: short random(short, short).

1.19 Ввод и вывод данных

* Язык предусматривает вывод данных в консоль ОС Windows.
* Для этого применяется оператор – write.
* Пример применения: write 123; write prt; write ‘sdgf’.
* После ключевого слова write следует переменная, литерал, вызов функции, заканчивается инструкция точкой с запятой.

1.20 Точка входа

* Точкой входа в программу является функция main.
* Шаблон:

main

{

Набор инструкций;  
}

* Программа обязательно содержит одну точку входа.

1.21 Препроцессор

* Язык содержит реализацию препроцессора.
* Препроцессор просматривает исходный код на наличии своих директив и при нахождении заменяет их необходимыми конструкциями языка.
* Директивы препроцессора начинаются с символа ‘#’.
* #require ‘Имя файла’ - заменяется на содержимое в файла, указанного в кавычках.

1.22 Соглашения о вызовах

* Используется соглашение о вызовах \_cdecl.
* Параметры передаются через стек.
* Параметры следуют справа налево.
* Стек очищается вызывающим кодом.

1.23 Объектный код

* Целевым языком трансляции - язык ассемблера для процессоров Intel семейства IA-32.

1.24 Классификация сообщений транслятора

1.25 Контрольный пример