# **Глава 1. Спецификация языка программирования**

## **Характеристика языка программирования**

Язык программирования GMP-2019 предназначен для выполнения сравнения целых чисел и операций над строками.

Язык программирования GMP-2019 является процедурным, строго типизированным, не объектно-ориентированным, компилируемым.

* 1. **Алфавит языка**

Алфавит языка GMP-2019 основан на кодировке ASCII. Таблица кодировки ASCII представлена на рисунке 1.1.

Исходный код может содержать символы латинского алфавита [a-z][A-Z], цифры десятичной системы счисления [0-9], символы сепараторы “, ; {}()‘ ’ (пробел)”, символы для операторов “<, >, =, !”. русские символы [а-я][А-Я] разрешены только в строковых литералах.

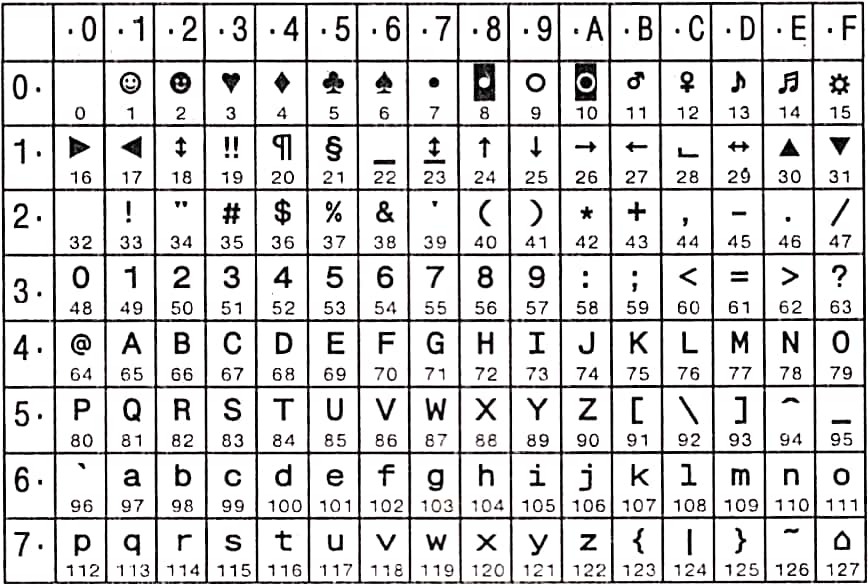


Рисунок 1.1 – Таблица кодировки ASCII

* 1. **Применяемые сепараторы**

Применяемые сепараторы в языке GMP-2019, приведены в таблице 1.1. Таблица 1.1 — Применяемые сепараторы

|  |  |
| --- | --- |
| Сепаратор | Назначение сепаратора |
| ; | Разделитель инструкций |
| { } | Программный блок |
| ( ) | Параметры |
| ‘ ’ | Служит для разделения. Допускается везде, кроме идентификаторов и ключевых слов |
| , | Разделитель параметров в функции |

* 1. **Применяемые кодировки**

Для написания исходного кода на языке программирования GMP-2019 используется кодировка ASCII, которая представлена на рисунке 1.1.

## **Типы данных**

В языке GMP-2019 есть 2 типа данных: целочисленный и строковый. Описание типов данных, предусмотренных в данным языке представлено в таблице 1.2.

Таблица 1.2 – Типы данных языка GMP-2019

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| integer | Фундаментальный тип данных. Предусмотрен для объявления беззнаковых целочисленных данных (2 байта).  Автоматически инициализируется нулевым значением.  Операции:  <= бинарная операция сравнения меньше либо равно;  >= бинарная операция сравнения больше либо равно;  == бинарная операция сравнения на равенство;  != бинарная операция сравнения на не равенство;  < бинарная операция сравнения меньше;  > бинарная операция сравнения больше; |
| line | Является строковым типом данных. Предназначен для работы с символами, каждый символ занимает 1 байт. Максимальное количество символов – 255, включая символ окончания строки.  Инициализация по умолчанию: нулевой символ. |

## **Преобразование типов данных**

Язык GMP-2019 строго типизированный, преобразование типов данных не поддерживается.

## **Идентификаторы**

Общее количество идентификаторов ограничено максимальным размером таблицы идентификаторов. Идентификаторы должны начинаться только с символов латинского алфавита, могут содержать цифры. Максимальная длина идентификатора равна 16 символам. Идентификаторы, объявленные внутри функционального блока, получают префикс, идентичный имени функции, внутри которой они объявлены. Префикс занимает 8 дополнительных символов. В случае превышения заданной длины, идентификаторы усекаются до длины, равной 16 символов (8 символов на имя идентификатора, 8 символов на префикс). Данные правила действуют для всех типов идентификаторов.(решулярные,пример)

## **Литералы**

В языке существует 2 типа литералов: целого и символьного типов. Краткое описание литералов представлено в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |
| --- | --- |
| Тип литерала | Описание |
| Литералы целого типа | Целочисленные беззнаковые литералы, десятичное и шестнадцатеричное представления. Литералы только rvalue.(0x..) |
| Строковые литералы | Состоит из символов, заключенных в "…" (двойные кавычки). Только rvalue. |

## **Объявление данных и область видимости**

В языке GMP-2019 объявление данных начинается с ключевого слова var, указывается тип данных и имя идентификатора. Требуется обязательное объявление переменной перед её использованием.

Примеры: var integer a, var line b;

Все переменные в языке GMP-2019 имеют область видимости, а именно префикс — название функции, в которой они находятся, что разрешает использование в различных функциях переменных с одинаковым именем. Параметры функции видны только внутри неё. Переменные, объявленные в одной функции, недоступны в другой.

## **Инициализация данных**

В момент объявления переменных в языке GMP-2019 происходит автоматическая инициализация в зависимости от типа данных. Инициализация другими значениями в момент объявления не допускается. Присвоение значения переменным должны происходить после непосредственного объявления. Виды инициализации представлены в таблице 1.4.

Таблица 1.4 — Способы инициализации переменных

|  |  |
| --- | --- |
| Вид инициализации | Примечание |
| var <тип данных> <идентификатор> = ..; | Автоматическая инициализация: переменные типа integer инициализируются нулём, переменные типа line – пустой строкой.(..=) |
| <идентификатор> = <значение>; | Присваивание переменной значения. |

## **Инструкции языка**

Инструкции языка GMP-2019 представлена в таблице 1.5.

Таблица 1.5 – Инструкции языка

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Объявление переменной | var <тип данных> <идентификатор>. |
| Объявление функции | <тип данных> function <идентификатор> (<тип данных> <идентификатор>, …)  {  / программный блок /  return <идентификатор/литерал>.  } |
| Вызов функции | <идентификатор> (<идентификатор>, ...)(бнф) |
| Присвоение значения | <идентификатор> = <значение>; |
| Печать данных | print <литерал/идентификатор>. |
| Функции стандартной библиотеки  (применяются при инициализации и в выражениях) | compare(line, line) — лексикографическое сравнение строк  pow(integer, integer) — возводит первый операнд типа integer в степень, равную значению второго операнда типа integer. (перенести) |
| Возвращаемое значение | return <литерал/идентификатор>. |

## **Операции языка**

Операции сравнения, при верности возвращают 1, иначе 0, которые можно использовать в языке GMP-2019, представлены в таблице 1.6.

Таблица 1.6 — Операции языка

|  |  |
| --- | --- |
| Инструкция | Форма записи |
| Операции сравнения языка | > — бинарное больше  < — бинарное меньше  >= — бинарное больше либо равно  <= — бинарное меньше либо равно  == — бинарное равно  != — бинарное не равно |

## **Выражения и их вычисление**

В языке присутствуют выражения сравнения. При истинности выражения возвращается 1, при ложности 0.(типы…)

## **Программные конструкции языка**

Основные программные конструкции языка GMP-2019 представлены в таблице 1.7.

Таблица 1.7 — Основные конструкции языка

|  |  |
| --- | --- |
| Конструкция | Реализация |
| Главная функция  (точка входа) | main  {  / программный блок /  } |
| Функции | <тип данных> function <идентификатор>  (<тип данных> <идентификатор>, …)  {  / программный блок /  return <идентификатор/литерал>.  } |
| Условный оператор | if(<истинное выражение>)  {  / программный блок /  }  else  {  / программный блок /  } |

## **Область видимости идентификаторов**

Область видимости «сверху вниз» (по принципу С++). В GMP-2019 требуется обязательное объявление переменной перед её использованием. Все переменные должны находиться внутри программного блока языка. Имеется возможность объявления одинаковых переменных в разных блоках. Каждая переменная получает префикс – название функции, в которой она объявлена

## **Семантические проверки**

В языке GMP-2019 выполняются следующие семантические проверки:

* проверка на присутствие единственной главной функции;
* проверка на правильную инициализацию;
* проверка на предварительное объявление идентификатора с ключевым словом var;
* проверка на дублирование идентификаторов;
* проверка на совпадение количества параметров функции;
* проверка на соответствие типа возвращаемого значения функции.

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

В языке предусмотрена секция данных и секция кода. Секция данных делится на секцию значений, секцию переменных и стек вычислений.

## **Стандартная библиотека и её состав**

В языке GMP-2019 предусмотрена стандартная библиотека. Функции, входящие в состав библиотеки, описаны в табл. 1.7. Стандартная библиотека подключается автоматически на этапе генерации кода.

Таблица 1.7 - Функции стандартной библиотеки языка GMP-2019

|  |  |
| --- | --- |
| Функция | Описание |
| compare(line a, line b) | Данная функция целочисленного типа, принимает два строковых параметра. Сравнивает строки и возвращает 0, если равны, -1, если a меньше b, 1, если a больше b |
| pow(integer a, integer b) | Данная функция целочисленного типа принимает два целочисленных параметра. Функция возводит число a в степень b и возвращает результат. |

## **Ввод и вывод данных**

Ввод данных языком программирования GMP-2019 не поддерживается.

Для вывода данных используется функция print (<имя идентификатора>).

Пример: print(a);

## **Точка** **входа**

Функция точки входа в языке программирования GMP-2019 представлена в таблице 1.10.

Таблица 1.10 — Точка входа

|  |  |
| --- | --- |
| Конструкция | Реализация |
| Главная функция  (точка входа) | main  {  / программный блок /  } |

## **Препроцессор**

Препроцессор, принимающий и выдающий некоторые данные на вход транслятору, в языке GMP-2019 отсутствует.

## **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**

GMP-2019 транслируется в язык ассемблера.

## **Классификация сообщений транслятора**

Генерируемые транслятором сообщения определяют степень его информативности, то есть сообщения транслятора должны давать максимально полную информацию о допущенной пользователем ошибке при написании программы. Сообщения транслятора приведены в таблице 1.10, а также в приложении А.

Таблица 1.10 Классификация ошибок(диапазон)

|  |  |
| --- | --- |
| Префикс ошибки | Описание ошибки |
| LEX-### | ### - код ошибки. Сообщение, генерируемое на этапе лексического анализа. |
| SYN-### | ### - код ошибки. Сообщение, генерируемое на этапе синтаксического анализа. |
| SEM-### | ### - код ошибки. Сообщение, генерируемое на этапе семантического анализа. |
| SYSTEM-### | ### - код ошибки. Сообщение, генерируемое при критической ошибке. |

## **Контрольный пример**

main

{

var line name1;

var line name2;

var integer a;

var integer b;

var integer result;

b = 0x0010;

name1 = "Mikhail";

name2 = name1;

result = compare(name1, name2);

if(result)

{

a = pow(2, 4);

}

else

{

a = 4;

}

if(a >= b)

{

print(b);

}

else

{

print(a);

}

}