МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6-05-0612-01 Программная инженерия

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора KVV-2024»

Выполнил студент Качинскас Вацловас Вацловович

(Ф.И.О.)

Руководитель проекта Наркевич Аделина Сергеевна

Заведующий кафедрой к.т.н., доц. Смелов В.В.

Консультанты Наркевич Аделина Сергеевна

Нормоконтролер Наркевич Аделина Сергеевна

Курсовой проект защищен с оценкой

Минск 2024

Содержание

Оглавление

[1. Спецификация языка программирования 3](#_Toc181789081)

[1.1. Характеристика языка программирования 3](#_Toc181789082)

# 1. Спецификация языка программирования

## 1.1. Характеристика языка программирования

Разрабатываемый язык программирования KVV-2024 – компилируемый, высокоуровневый, императивный, строго типизированный, явный.

## 1.2. Определение алфавита языка программирования

Язык программирования KVV-2024 использует стандартную кодировку символов Windows-1251. Его структура представлена в форме Бэкуса-Наура в таблице 1.1.

|  |
| --- |
| <строчная буква латинского алфавита>::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z |
| <прописная буква латинского алфавита>::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z |
| <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| <символ- сепаратор>::= ' '|,|(|)|{|}|; |
| <побитовые операция>::= | | & | ! |

Таблица 1.1 – Алфавит языка программирования KVV-20K24

## 1.3. Применяемые сепараторы

Сепараторы являются неотъемлемой частью языка программирования. Они используются для логического разделения конструкций, функций и лексем.

Сепараторы, используемые в языке программирования KVV-2024, и их назначение представлены в таблице 1.2.

|  |  |
| --- | --- |
| Сепаратор | Назначение |
| пробел | Разделитель лексем. |
| ; (точка с запятой) | Разделитель строк |
| {} | Определитель блока фукнций |
| () | Определитель параметров фукнций |
| , (запятая) | Разделитель параметров фукнции |
| ‘’ | Определение символьного литерала |

Таблица 1.2 – Применяемые сепараторы KVV-2024

## 1.4. Применяемые кодировки

Для написания исходного кода программы на языке KVV-2024 используется кодировка Windows-1251. Русские символы разрешается использоваться только в символьных литералах.

## 1.5. Типы данных

Язык программирования KVV-2024 поддерживает использование следующих типов данных: беззнаковый целый, символьный. Описание типов данных представлено в таблице 1.3.

|  |  |
| --- | --- |
| Тип данных | Описание |
| Беззнаковый целый | Беззнаковый целочисленный тип данных. Занимает 4 байта. Диапазон значений: от 0 до 4294967295. Инициализация по умолчанию: 0. |
| Символьный | Символьный тип данных. символ занимает 1 байт. Диапазон значений: все символы кодировки Windows-1251. Инициализация по умолчанию: ‘\0’ (нулевой байт). |

Таблица 1.3. – Типы данных KVV-2024

## 1.6. Преобразование типов данных

Символьный тип данных может быть преобразован в беззнаковый целый в виде кода символа, и наоборот.

## 1.7. Идентификаторы

Идентификатор языка – это последовательность символов, используемая в качестве имени для идентификации сущностей в языке программирования.

В KVV-2024 идентификаторы используются для именования переменных, функций и их параметров.

Формальное описание идентификатора представлено в форме Бэкуса-Наура в таблице 1.4

|  |
| --- |
| <идентификатор> ::= <буква> | '\_' {<буква> | <цифра> | '\_'} |
| <буква> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| <цифра> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Таблица 1.4 – Идентификаторы KVV-2024

В языке KVV-2024 идентификаторы не могут содержать символов кириллицы, специальных символов за исключением нижнего подчеркивания; не могут начинаться с цифры.

Примеры правильных идентиификаторов: s0mthing, name, can\_get.

Примеры неправильных идентификаторов: 0var, somth!ng.

## 1.8. Литералы

Литералы языка программирования – последовательность символов, используемая для именования неизменяемых значений в коде.

В языке KVV-2024 предусмотрены следующие типы литералов: целочисленный, символьный. Описание литералов представлено в таблице 1.5.

|  |  |
| --- | --- |
| Литерал | Описание |
| Беззнаковый целый | Могут состоять только из цифр[0..9] без дробной части и без знака.  При выходе за пределы допустимости будет выведена соответствующая ошибка.  Произвольная система счисления |
| Символьный | Состоит из одного символа, заключенного в ‘’ (одинарные кавычки). |
| VOID, BOOL |  |

Таблица 1.5. – Литералы KVV-2024

*Пример правильных литералов: 2024, ‘V’.*

*Пример неправильных литералов: k, -4.*

## 1.9. Объявление данных

При объявлении переменных в языке KVV-2024 необходимо учитывать следующие правила:

* каждая переменная должна быть объявлена до её использования.
* объявление начинается с типа переменной, за которым следует идентификатор.
* тип переменной должен быть одним из допустимых в KVV-2024 типов.

Объявление функции схоже с объявлением обычной переменной:

* Каждыя функция должна быть объявлена перед применением
* Объявление и определение начинается с ключевого слова “function”, за которым следует тип возвращаемого значения с идентификатором, после в скобках пишутся параметры

## 1.10. Инициализация данных

Во время объявления переменной можно присвоить ей значение с помощью оператора присваивания (=). По умолчанию, переменным различных типов данных присваиваются начальные значения.

## 1.11. Инструкции языка

Инструкции языка программирования KVV-2024 представлены в таблице 1.6.

|  |  |  |
| --- | --- | --- |
| Инструкция | Описание | Пример |
| Объявление переменной | <тип данных> <идентификатор>; | uint number;  char symbol; |
| Инициализация переменной | <тип данных> <идентификатор> = <литерал>| <идентификатор> | uint number = 14;  char symbol = number; |
| Объявление фукнции | function <тип данных> <идентификатор функции> (<список параметров>) {<блок функции>}; | function uint sum(int a, char b) {}; |
| Выход из функции | return <идентификатор> | <литерал>; | return number;  return ‘V’; |
| Присваивание | <идентификатор> = <литерал>;  <идентификатор> = <идентификатор>; | number = 625;  number = symbol; |
| Вывод данных | write(<идентификатор> | <литерал>);  writeLine(<идентификатор> | <литерал>); | write(number)  writeLine(‘j’); |
| Ввод данных | read(<идентификатор> | <void>); | read(number); |

Таблица 1.6 – Инструкции языка KVV-2024

## 1.12. Операции языка

Язык программирования KVV-2024 поддерживает различные типы операций. Их описание, приоритетность выполнения представлены в таблице 1.7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Операция | Арифметическое назначение | Приоритетность операции | Свойства | Количество операндов | Тип операндов | Тип результата | Порядок выполнения |
| | | Побитовое или | 3 | Коммутативность, дистрибутивность | 2 | Целочисленные беззнаковые | Целочисленный беззнаковый | Слева-направо |
| & | Побитовое и | 2 | Коммутативность, дистрибутивность | 2 | Целочисленные беззнаковые | Целочисленный беззнаковый | Слева-направо |
| ! | Побитовая инверсия | 1 | Ассоциативность, коммутативность, дистрибутивность | 1 | Целочисленные беззнаковые | Целочисленный беззнаковый | Слева-направо |
| ^ | Исключающее ИЛИ | 2 | Коммутативность | 2 | Целочисленные беззнаковые | Целочисленный беззнаковый | Слева-направо |
| ЛОГИЧЕСКИ |  |  |  |  |  |  |  |

Таблица 1.7 – Операции KVV-2024

## 1.13. Выражения и их вычисление

Выражение в языке KVV-2024 представляет собой комбинацию операндов (переменных, констант, функций) и операций, которая вычисляется в определённом порядке, возвращая результат. Выражения могут включать вызовы функций, побитовые операции, условные конструкции и использование литералов.

Тип: Побитовые и условные выражения.

Правила составления выражений:

* Операции выполняются в порядке их приоритетности, который может быть изменён с помощью круглых скобок.
* Все операнды должны быть совместимы по типу с используемой операцией.
* Результат выражения имеет тип, соответствующий правилам приведения типов для конкретной операции.

Операции, используемые в выражениях:

* Побитовые операции: &, |, ^, ~, <<, >>.
* Логические операции: &&, ||, !.
* Условный оператор: ? :.
* Функции стандартной библиотеки (например, сравнение строк).

Допустимые типы операндов:

* Беззнаковый целый тип (uint).
* Символьный тип (char).
* Логический тип (bool).

Порядок вычисления подвыражений:

* Операции с более высоким приоритетом выполняются раньше операций с более низким приоритетом.
* Если приоритеты одинаковы, выражения вычисляются слева направо (за исключением унарных операций, выполняемых справа налево).
* Подвыражения в круглых скобках вычисляются в первую очередь.
* В логических выражениях используется короткое замыкание:
  + В операции && вычисление прекращается, если первый операнд равен false.
  + В операции || вычисление прекращается, если первый операнд равен true.

## 1.14 Конструкции языка

Конструкции языка KVV-2024 представлены в таблице 1.7.

|  |  |
| --- | --- |
| Конструкция | Представление в языке |
| Главная функция | main  {<инструкции языка>} |
| Функция | function <тип данных> <идентификатор>(<тип данных> <идентификатор>, …)  {  {<инструкции языка>}  return<идентификатор>|<литерал>;  } |
| Блок | {  …  } |

Таблица 1.7 - Конструкции языка

Исходный текст программы на языке программирования KVV-2023 оформляется в виде главной функции, точки входа в программу. При составлении функций рекомендуется выделять блоки и фрагменты отступами для лучшей читаемости кода.

## 1.15 Область видимости идентификаторов

Область видимости в языке KVV-2024 организована следующим образом: все идентификаторы доступны из текущей области видимости или из вложенных областей видимости. Области видимости выделяются {}.

## 1.16 Семантические проверки

В языке программирования KVV-2024 выполняются следующие семантические проверки:

* единственность точки входа;
* переопределение идентификаторов;
* использование идентификаторов без их объявления;
* проверка соответствия типа функции и возвращаемого параметра;
* правильность передаваемых в функцию параметров: количество, типы;
* правильность выражений;
* превышение размера литералов;

## 1.17 Распределение оперативной памяти на этапе выполнения

На этапе выполнения программы в языке KVV-2024 используется модель управления памятью, включающая несколько областей. Каждая область отвечает за хранение различных типов данных и структур. Управление памятью частично автоматизировано, но программист также может явным образом работать с памятью через вызовы функций стандартной библиотеки.

Виды областей памяти:

1. Область кода:
   * Содержит скомпилированный машинный код программы, в данном случае представленный ассемблерными инструкциями.
   * Является статической и неизменяемой во время выполнения.
2. Статическая область:
   * Хранит глобальные переменные и данные, которые существуют на протяжении всего времени выполнения программы.
   * Инициализация данных выполняется до начала выполнения основной программы.
3. Стек:
   * Используется для хранения локальных переменных, параметров функций и адресов возврата.
   * Упорядочен по принципу "последним пришёл — первым вышел" (LIFO).
   * Размер стека ограничен, и его переполнение приводит к ошибке выполнения.
4. Динамическая область:
   * Отведена для данных, размещаемых в процессе выполнения программы (например, через вызовы функций стандартной библиотеки).
   * Управление этой областью памяти возлагается на программиста: он должен явно выделять и освобождать память.
   * Возможны утечки памяти при неправильном управлении.

Эффективное использование этих областей памяти позволяет минимизировать ошибки выполнения и оптимизировать производительность программы.

## 1.18. Стандартная библиотека и ее состав

1. Сравнение строк:
   * Назначение: сравнивает две строки и возвращает результат их сравнения (меньше, равно, больше).
   * Параметры: две строки (char[]).
   * Возвращаемое значение: целое число (int), где -1 означает "меньше", 0 — "равны", 1 — "больше".
2. Произвольная функция:
   * Назначение: выполняет специфические задачи, определённые программистом.
   * Параметры: зависит от реализации.
   * Возвращаемое значение: определяется задачей функции.

Способ подключения и применение:

* Стандартная библиотека подключается автоматически компилятором.
* Вызов функций осуществляется через их имя с указанием необходимых параметров.

**1.19. Ввод и вывод данных**

**Операторы ввода и вывода**:

1. **Вывод данных**:
   * write: выводит данные без перехода на новую строку.
   * writeline: выводит данные с переходом на новую строку.
   * Применение:

writeline("Привет, мир!");

write("Введите число: ");

**Формальное описание операторов**:

* write(<значение>): выводит значение в стандартный поток вывода.
* writeline(<значение>): выводит значение в стандартный поток вывода, добавляя символ перевода строки.

**1.20. Точка входа**

**Определение точки входа**:

* Точка входа — это поименованная функция main, с которой начинается выполнение программы.

**Синтаксис**:

void main() {

// Тело программы

}

**1.21. Препроцессор**

**Наличие препроцессора**:

* В языке KVV-2024 отсутствует полноценный препроцессор.
* Вместо этого поддерживается минимальный набор директив (например, #include для стандартной библиотеки).

**1.22. Соглашения о вызовах**

**Описание**:

* Используется стандартное соглашение о вызовах для платформы Windows x64.
* Параметры передаются через регистры, остальные через стек.
* Результат возвращается через регистр RAX.

**1.23. Объектный код**

**Целевой язык трансляции**:

* Код компилируется в ассемблерный код для платформы x64.

**1.24. Классификация сообщений транслятора**

**Классы сообщений**:

1. Ошибки синтаксиса (код E001): "Ошибка в синтаксисе выражения".
2. Ошибки типов (код E002): "Несовместимость типов операндов".
3. Предупреждения (код W001): "Переменная объявлена, но не используется".

**1.25. Контрольный пример**

Программа демонстрирует основные возможности языка KVV-2024:

void main() {

uint x = 5;

uint y = 3;

uint result = (x & y) | (x << 1);

writeline("Результат: ", result);

}