МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 6-05-0612-01 “Программная инженерия”

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка компилятора AAM – 2024»

Выполнил студент Альшевская Алина Михайловна

(Ф.И.О.)

Руководитель проекта ст. преп. Наркевич А.С.

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

Заведующий кафедрой ст. преп. Наркевич А.С.

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

Консультанты ст. преп. Наркевич А.С.

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

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

Нормоконтролер ст. преп. Наркевич А.С.

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

Курсовой проект защищен с оценкой

Минск 2024

**Содержание**

[Введение 3](#_Toc122449900)

[1 Спецификация языка программирования 4](#_Toc122449901)

[1.1 Характеристика языка программирования 4](#_Toc122449902)

[1.2 Определение алфавит языка программирования 4](#_Toc122449903)

[1.3 Применяемые сепараторы 4](#_Toc122449904)

[1.4 Применяемые кодировки 5](#_Toc122449905)

[1.5 Типы данных 5](#_Toc122449906)

[1.6 Преобразование типов данных 6](#_Toc122449907)

[1.7 Идентификаторы 6](#_Toc122449908)

[1.8 Литералы 6](#_Toc122449909)

[1.9 Область видимости идентификаторов 7](#_Toc122449910)

[1.10 Инициализация данных 7](#_Toc122449911)

[1.11 Инструкции языка 7](#_Toc122449912)

[1.13 Выражения и их вычисления 9](#_Toc122449913)

[1.14 Программные конструкции языка 9](#_Toc122449914)

[1.15 Область видимости 9](#_Toc122449915)

[1.16 Семантические проверки 9](#_Toc122449916)

[1.17 Распределение оперативной памяти на этапе выполнения 10](#_Toc122449917)

[1.18 Стандартная библиотека и её состав 10](#_Toc122449918)

[1.19 Ввод и вывод данных 11](#_Toc122449919)

[1.20 Точка входа 11](#_Toc122449920)

[1.21 Препроцессор 11](#_Toc122449921)

[1.22 Соглашения о вызовах 12](#_Toc122449922)

[1.23 Объектный код 12](#_Toc122449923)

[1.24 Классификация сообщений транслятора 12](#_Toc122449924)

[1.25 Контрольный пример 12](#_Toc122449925)

# **Введение**

Задачей данного курсового проекта является разработка транслятора для своего языка программирования: AAM-2024.

Написание транслятора будет осуществляться на языке C++, при этом код на языке AAM-2024 будет транслироваться в язык ассемблера.

Транслятор AAM-2024 состоит из следующих частей:

– семантический анализатор;

– синтаксический анализатор;

– логический анализатор;

– генератор исходного кода на языке ассемблера.

Исходя из цели курсового проекта, были определены следующие задачи:

– разбработка спецификации языка программирования;

– разбработка структуры транслятора;

– разработка лексического и семантического анализаторов;

– разработка синтаксического анализатора;

– преобразование выражений;

– генерация кода на язык ассемблера;

– тестирование транслятора.

Решения каждой из поставленных задач будут приведены в соответствующих главах курсового проекта.

# **1 Спецификация языка программирования**

## **Характеристика языка программирования**

Язык программирования AAM-2024 – это универсальный язык высокого уровня. Он является процедурным, компилируемым, не объектно-ориентированным. Язык строго типизируемый, что говорит о невозможности преобразования типов, транслируемым языком программирования.

## **Определение алфавит языка программирования**

Алфавит языка программирования – это набор символов, которые используются для написания компьютерных программ. На этапе выполнения могут использоваться символы латинского алфавита, цифры десятичной системы счисления от 0 до 9, спецсимволы, а также непечатные символы пробела, табуляции и перевода строки. Русские символы разрешены только в символьных литералах.

Таблица 1.1 – Алфавит языка программирования AAM-2024

|  |
| --- |
| <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>::= ' '|,|(|)|{|}|;|”|’|=|+|-|\*|| |

## **Применяемые сепараторы**

Символы-сепараторы служат в качестве разделителей операций языка. Сепараторы, используемые в языке программирования AAM-2024, приведены в таблице 1.1.

Таблица 1.1 – Сепараторы

|  |  |  |
| --- | --- | --- |
| Сепаратор | Название | Область применения |
| ‘ ‘ | Пробел | Допускается везде, кроме идентификаторов и ключевых слов |
| ; | Точка с запятой | Разделение конструкций |
| {…} | Фигурные скобки | Заключение программного блока |
| […] | Квадратные кавычки | Блок кода |
| (…) | Круглые скобки | Приоритет операций, параметры функции |
| “…” | Двойные кавычки | Строковый литерал |
| ‘…’ | Одинарные кавычки | Допускается везде, кроме идентификаторов и ключевых слов |
| = | Знак «равно» | Присваивание значения |
| , | Запятая | Разделение параметров |
| +  -  \*  / | Знак «плюс»,  знак «минус»,  знак «астерикс»,  знак «обратная косая черта» | Знаки математических операций. Допускаются только в математических операциях, между идентификаторами или числовыми константами. |
| ==  !  <  > | Двойное «равно»,  «восклицательный знак»,  знак «меньше»,  знак «больше» | Допускаются в логических или тернарных операторах. |

## **Применяемые кодировки**

Для написания исходного кода на языке программирования AAM-2024 используется кодировка Windows-1251.

Содержимое таблицы символов представлено на рисунке 1.1.



Рисунок 1.1 – Таблица кодировки Windows-1251

## **Типы данных**

В языке AAM-2024 реализованы четыре типа данных: целочисленный (int), символьный(char), строковый(string) и логический (bool). Описание типов данных, предусмотренных в данным языке представлено в таблице 1.2.

Таблица 1.2 – Типы данных языка AAM-2024

|  |  |
| --- | --- |
| Тип данных | Описание типа данных |
| Целочисленный тип данных  int | Фундаментальный тип данных. Используется для работы с целочисленными значениями. Знаковый тип. В памяти занимает 4 байта.  При попытке инициализации значением больше максимального, инициализируется максимальным. При попытке инициализации значением меньше минимального, инициализируется минимальным.  Максимальное значение: 2147483647. Минимальное значение: -2147483647.  Применяемые операции:  + (бинарный) – сложение;  - (бинарный) – вычитание;  - (унарный) – отрицательное число;  \* (бинарный) – умножение;  / (бинарный) – деление;  % (бинарный) – остаток от деления; |
| Символьный тип данных char | Фундаментальный тип данных. Используется для работы с символом, который в памяти занимает 1 байт. Сохраняет символы из набора символов ASCII.  Допустимый диапазоны значений: от 0 до 255.  Применяемые операции:  + (бинарный) – конкатенация. |
| Строковый тип данных str | Фундаментальный тип данных. Используется для работы с набором символов типа char, каждый символ в памяти занимает 1 байт.  Максимальная длина строки 255 символа, последний всегда символ конца строки “\0”.  Занимает память в зависимости от количества символов, максимально может занимать 256 байт в памяти.  Применяемые операции:  + (бинарный) – конкатенация. |
| Логический тип данных  bool | Фундаментальный тип данных. Используемся для работы с логическими значениями: истина (true) и ложь (false). Переменная этого типа может иметь значения [true](https://learn.microsoft.com/ru-ru/cpp/cpp/true-cpp?view=msvc-170) и [false](https://learn.microsoft.com/ru-ru/cpp/cpp/false-cpp?view=msvc-170). Занимает в памяти 1 байт. |

Пользовательские типы данных не поддерживаются.

## **Преобразование типов данных**

Преобразование типов данных не поддерживается, т.е. язык является строго типизированным.

## **Идентификаторы**

В имени идентификатора допускаются только символы латинского алфавита и знак «\_» и цифры. Максимальная длина имени идентификатора – 8 символов. Максимальная длина имени идентификатора функции – 11 символов. При вводе идентификатора длиной более разрешенного количества символов, он будет усекаться. Имя идентификатора не может совпадать с именем функции, уже содержащаяся в стандартной библиотеке, если только это функция подключена через оператор extern.

## **Литералы**

С помощью литералов осуществляется инициализация переменных. В языке существует четыре типа литералов. Краткое описание литералов языка AAM-2024 представлено в таблице 1.3.

Таблица 1.3 – Описание литералов

|  |  |  |  |
| --- | --- | --- | --- |
| Тип литерала | Регулярное выражение | Описание | Пример |
| Целочисленный литерал | -?[1-9]+[0-9]\*- для десятичной системы  0[x|X][0-9| a-f|A-F]{2} – для шестнадцатеричной системы | Целочисленные литералы не имеют дробных частей или экпонент. Представлены в десятичной и шестнадцатеричной системах счисления. В шестнадцатеричной системе счисления символы a-f (A-F) представляют числа от 10 до 15 соответственно. Шестнадцатеричные литеры начинаются с префикса 0x или 0Х . | create int sum = 15;  15 – целочисленный литерал.  sum=0x0F;  0x0F -целочисленный литерал |
| Символьный литерал | [a-z|A-Z|А-Я|а-я|0-9|!-/] | Символ, заключённый в '' (одинарные кавычки). Литералы могут быть только rvalue. | create char symbol = 'T';  T – символьный литерал. |
| Строковый литерал | [a-z|A-Z|А-Я|а-я|0-9|!-/]\* | Набор символьных литералов, заключенных в '' (одинарные кавычки) или ""(двойные кавычки). Максимальная длина строки 255 символов. | write “text”;  text - строковый литерал. |
| Логический Литерал | [true|false] | Логический литерал имеет два возможных значения: true (истина) и false (ложь) | create bool check = true;  true – логический литерал |

Литералы являются константами и при генерации кода объявляются один раз.

## **Область видимости идентификаторов**

Область видимости «сверху вниз». В языке AAM-2024 требуется обязательное объявление переменной перед её инициализацией и последующим использованием. Все переменные должны находиться внутри программного блока. Имеется возможность объявления одинаковых переменных в разных блоках, т. к. переменные, объявленные в одной функции, недоступны в другой. Каждая переменная получает префикс – название функции, в которой она объявлена. Объявление функций стандартной библиотеки можно производить в любом месте кода.

## **Инициализация данных**

При объявлении переменной не допускается инициализация. Описание способов инициализации переменных языка AAM-2024 представлено в таблице 1.4.

Таблица 1.4 – Способы инициализации переменных

|  |  |  |
| --- | --- | --- |
| Конструкция | Описание | Пример |
| create <тип данных> <идентификатор>; | Автоматическая инициализация: переменные типа int инициализируются нулём, переменные типа char – пустым символом. | create int sum;  create char chr; |
| <идентификатор> = <значение>; | Присваивание переменной значения. | sum = 9;  chr = ‘D’; |

Соответствие типов проверяется на синтаксическом анализе.

## **Инструкции языка**

Все возможные инструкции языка программирования AAM-2024 представлены в общем виде в таблице 1.5.

Таблица 1.5 – Инструкции языка программирования AAM-2024

|  |  |
| --- | --- |
| Инструкция | Запись на языке AAM-2024 |
| Объявление переменной | create <тип данных> <идентификатор>; |
| Объявление функции | create <тип данных> func <идентификатор> (<тип данных> <идентификатор>, …) {<блок кода>}; |
| Присваивание | <идентификатор> = <значение>/<идентификатор>; |
| Объявление внешней функции | connect <тип данных> func <идентификатор> (<тип данных> <идентификатор>, …); |
| Блок инструкций | {  …  } |

Продолжение таблицы 1.5

|  |  |
| --- | --- |
| Инструкция | Запись на языке KNP-2024 |
| Возврат из подпрограммы | return <идентификатор> / <литерал>; |
| Условная инструкция | while(<условие>)[<блок кода>]; |
| Вывод данных | write <идентификатор> / <литерал>;  writeline <идентификатор> / <литерал>; |
| Однострочный комментарий до конца строки | #<любой текст> |

Инструкции (кроме функции входа в программу) требуют закрывающую «;».

* 1. **Операции языка**

Язык программирования AAM-2024 может выполнять операции, представленные в таблице 1.6. Операция сдвига учитывает только первый младший бит оператора, т.к. сдвиг более чем на 255 любого числа кроме нуля вернет число большее, чем можно разместить в типе данных int.

Таблица 1.6 – Операции языка программирования KNP-2024

|  |  |  |  |
| --- | --- | --- | --- |
| Операция | Примечание | Типы данных | Пример |
| ( | Приоритет операций | - | sum = (a + b) \* c; |
| ) |
| + | Суммирование | (int, int)  (char, char)  (str, str) | sum = a + b; |
| - | Вычитание | (int, int) | diff = a – b; |
| \* | Умножение | (int, int) | mul = a \* b; |
| : | Деление | (int, int) | div = a : b; |
| % | Остаток от деления | (int, int) | mod = a % b; |
| & | Поразрядное «и» | (int, int)  (bool, bool) | pr = a & b; |
| | | Поразрядное «или» | (int, int)  (bool, bool) | res = a | b; |
| = | Присваивание | (int, int)  (char, char)  (str, str)  (bool,bool) | sum = 15;  chr = ‘T’; |
| <,> | Знаки «больше», «меньше» для условной инструкции | (int, int)  (char, char)  (str, str) | while(sum < diff) […]; |
| == | Оператор эквивалентности | (int, int)  (char, char)  (str, str) | while(sum == diff) […]; |
| ! | Оператор неравенства | (int, int)  (char, char)  (str, str) | while(sum ! diff) […]; |

## **Выражения и их вычисления**

Круглые скобки в выражении используются для изменения приоритета операций. Также не допускается запись двух подряд идущих арифметических операций. Выражение может содержать вызов функции, если эта функция уже содержится в стандартной библиотеке. Выражения вычисляются только после оператора присваивания.

## **Программные конструкции языка**

Ключевые программные конструкции языка программирования AAM-2024 представлены в таблице 1.7.

Таблица 1.7 – Программные конструкции языка AAM-2024

|  |  |
| --- | --- |
| Конструкция | Запись на языке AAM-2024 |
| Главная функция (точка входа) | main  {  …  return <идентификатор> / <литерал>;  } |
| Функция | create <тип данных> func <идентификатор> (<тип> <идентификатор>, …)  {… return <идентификатор> / <литерал>; }; |
| Цикл | while(a!8)[ …]; |
| Условный оператор | if(5>4)[…]; |

Программные конструкции языка AAM-2024 представляют собой базовый функционал для выполнения различных операций, что делает возможным решать задачи различного уровня.

## **Область видимости**

В языке AAM-2024 все переменные являются локальными, т.е. имеют функциональную область видимости. Они обязаны находится внутри программного блока функций . Объявление глобальных переменных не предусмотрено. Объявление пользовательских областей видимости не предусмотрено.

## **Семантические проверки**

Назначение семантического анализа – проверка смысловой правильности конструкций языка программирования. Таблица с перечнем семантических проверок, предусмотренных языком, приведена в таблице 1.8.

Таблица 1.8 – Семантические проверки

|  |  |
| --- | --- |
| Номер | Правило |
| 1 | Идентификаторы не должны повторно объявляться в пределах одной функции. |
| 2 | Тип возвращаемого значения должен совпадать с типом функции при её объявлении или подключении |
| 3 | Тип данных передаваемых значений в функцию должен совпадать с типом параметров при её объявлении или подключении |
| 4 | В функцию должно быть передано то число параметров, сколько ожидается |
| 5 | Тип данных результата выражения должен совпадать с типом данных идентификатора, которому оно присваивается |
| 6 | Типы данных операндов выражения должны быть одинаковыми |
| 7 | Тип данных string не может быть аргументом условной конструкции |
| 8 | Для типа char определены только операция + |
| 9 | Функции не должны подключаться дважды в пределах одной программы |

Если семантическая проверка не проходит, то в лог журнал записывается соответствующая ошибка.

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

Все переменные размещаются в стеке.

Стек — это структура данных, организованная по принципу LIFO (последний вошел - первый вышел). Данное решение идеально подходит для хранения данных, к которым вскоре предстоит обратиться (легко извлекаются с вершины стека).

## **Стандартная библиотека и её состав**

Стандартная библиотека AAM-2024 написана на языке программирования C++.

Для использования функций стандартной библиотеки, нужно явно подключить необходимую функцию с помощью ключевого слова connect, далее работа с ними производится как с пользовательскими функциями. Функции стандартной библиотеки с описанием представлены в таблице 1.9.

Таблица 1.9 – Состав стандартной библиотеки

|  |  |  |
| --- | --- | --- |
| Функция(C++) | Возвращаемое значение | Описание |
| int GetSize(string text) | int | Возвращает длину строки |
| int StrToInt(string text) | int | Преобразует строку в число |

Так же в библиотеке присутствуют приватные функции. Их описание представлено в таблице 1.10.

Таблица 1.10 – Приватные функции стандартной библиотеки

|  |  |  |
| --- | --- | --- |
| Функция(C++) | Возвращаемое значение | Описание |
| void writeint (int a) | – | Выводит число на экран Вызывается оператором write |
| void writechar (char a) | – | Выводит символ на экран Вызывается оператором write |
| void writestring (void\* in) | – | Выводит строку на экран Вызывается оператором write |

Приватные функции не могут быть вызваны явно и не требуют предварительного пользовательского подключения. Они вызываются специальными операторами языка.

## **Ввод и вывод данных**

В языке AAM-2024 не реализованы средства ввода данных.

Для вывода данных в стандартный поток вывода предусмотрен оператор write и writeline, который базируется на приватных функциях стандартной библиотеки.

## **Точка входа**

В языке AAM-2024 каждая программа должна содержать главную функцию main, т. е. точку входа, с которой начнется последовательное выполнение программы.

## **Препроцессор**

Препроцессор в языке программирования AAM-2024 не предусмотрен.

## **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**

AAM-2024 транслируется в язык ассемблера.

## **Классификация сообщений транслятора**

В случае возникновения ошибки в коде программы на языке AAM-2024 и выявления её транслятором в текущий файл протокола выводится сообщение. Классификация сообщений приведена в таблице 1.10.

Таблица 1.10. – Классификация сообщений транслятора

|  |  |
| --- | --- |
| Интервал | Описание ошибок |
| 0-99 | Системные ошибки |
| 100-109 | Ошибки параметров |
| 110-119 | Ошибки открытия и чтения файлов |
| 120-139 | Ошибки лексического анализа |
| 600-699 | Ошибки синтаксического анализа |
| 700-900 | Ошибки семантического анализа |

Компилятор может обрабатывать до 1000 различных ошибок.

## **Контрольный пример**

Код контрольного примера представлен в Приложении А.