ГОУ ВПО Академия государственной службы

Кафедра информатики и математики

Контрольная работа

По теме « Языки программирования: составные части и назначения"  
По дисциплине: Информатика

Проверила: Геращенко М. М.  
Выполнила: Кошман Юлия Владимировна,  
группа 10132  
  
  
  
  
  
  
  
  
  
Новосибирск 2010

Оглавление.

[Введение. - 3 -](#_Toc275697615)

[Что такое программа и язык программирования? - 3 -](#_Toc275697616)

[Этапы разработки программы. - 4 -](#_Toc275697617)

[Состав языка. - 4 -](#_Toc275697618)

[А) Операторы. - 4 -](#_Toc275697619)

[Б) Алфавит и лексемы. - 5 -](#_Toc275697620)

[В) Константы. - 5 -](#_Toc275697621)

[Г) Имена, ключевые слова и знаки операций. - 6 -](#_Toc275697622)

[Д) Переменные. - 7 -](#_Toc275697623)

[Е) Пример программы. - 7 -](#_Toc275697624)

[Заключение. - 8 -](#_Toc275697625)

[Приложение №1(константы). - 9 -](#_Toc275697626)

[Приложение №2 (ключевые слова). - 10 -](#_Toc275697627)

[Приложение №3( языки программирования). - 11 -](#_Toc275697628)

[Приложение №4 (специальные символы). - 12 -](#_Toc275697629)

[Приложение №5 ( составные символы). - 13 -](#_Toc275697630)

[Список литературы - 14 -](#_Toc275697631)

# Введение.

По моему мнению, тема « ЯЗЫКИ ПРОГРАММИРОВАНИЯ» очень актуальна на сегодняшний день, ведь прогресс компьютерных технологий определил процесс появления новых разнообразных знаковых систем для записи алгоритмов, для повышения эффективности труда пользователя, а также сокращения времени и затрат на подготовку и выполнение программ.

Команды, выполняемые процессором, являются электрическими сигналами, которые можно представить в виде нулей и единиц. Каждой команде соответствует свое число. Таким образом, процессор имеет дело с машинным кодом. Написать программу на нем может только очень опытный программист, хорошо знающий "архитектуру" процессора ( его устройство) и систему команд ( набор допустимых инструкций).

Совокупность средств и правил представления алгоритма в виде, пригодном для выполнения вычислительной машиной, называется языком программирования. Из чего же состоят эти языки программирования, какого их назначения и примеры, я рассмотрю в своем реферате.

# Что такое программа и язык программирования?

**Программа** - это запись (реализация) алгоритма на языке программирования.

"...Программа - законченный последовательный набор команд, операторов, реализующих решение функциональной задачи..."

**Извлечение из документа:**

Приказ Минздравсоцразвития РФ от 16.01.2006 N 22  
"Об утверждении Межотраслевых типовых норм времени на работы по сервисному обслуживанию оборудования телемеханики, сопровождению и доработке программного обеспечения"

**Язык** – это набор правил, определяющих систему записей, составляющих программу.   
**Язык программирования** - это специальный язык, на котором пишут команды для управления компьютером. Языки программирования созданы для того, чтобы людям было проще читать и писать для компьютера, но они затем должны транслироваться в машинный код, который только и может исполняться компьютером.   
Языки программирования можно разделить на языки высокого уровня и языки низкого уровня.

**Язык низкого уровня** - это язык программирования, предназначенный для определенного типа компьютера и отражающий его внутренний машинный код; языки низкого уровня часто называют машинно-ориентированными языками. Их довольно сложно изучать, поскольку для этого требуется хорошо знать принципы внутренней работы компьютера. Примеры (ассемблер, форт).

**Язык высокого уровня** - это язык программирования, предназначенный для удовлетворения требований программиста; он не зависит от внутренних машинных кодов компьютера любого типа. Языки высокого уровня используют для решения проблем и поэтому их часто называют проблемно-ориентированными языками. Каждая команда языка высокого уровня эквивалентна нескольким командам в машинных кодах, поэтому программы, написанные на языках высокого уровня, более компактны, чем аналогичные программы в машинных кодах.

# Этапы разработки программы.

В процессе создания любой программы можно выделить несколько этапов.

* Постановка задачи – выполняется специалистом. Необходимо определить цель задачи, ее содержание и общий подход к решению.
* Анализ задачи и моделирование – определяются исходные данные и результат.
* Разработка или выбор алгоритма решения задачи – программист должен выбрать оптимальное решение на основе математического описания.
* Проектирование общей структуры программы – формируется модель решения с последующей детализацией.
* Кодирование – запись алгоритма на языке программирования.
* Откладка и тестирование программы. Под откладкой понимается устранение ошибок в программе. Тестирование позволяет вести их поиск и, в конечном итоге, убедиться в том, что полностью отлаженная программа дает правильный результат.
* Анализ результатов.

# Состав языка.

# А) Операторы.

Язык программирования можно уподобить очень примитивному иностранному языку с жесткими правилами, не имеющими исключений. Изучение иностранного языка обычно начинают с алфавита, затем переходят к простым словам, далее рассматривают законы построения фраз, и только в результате длительной практики становится свободно выражать на этом языке свои мысли.

Программа состоит из исполняемых операторов и операторов описания.

**Исполняемый оператор** задает законченное действие, выполняемое над данными. Примеры исполняемых операторов: вывод на экран, занесение числа в память, выход из программы.

**Оператор описания**, как и следует из его названия, описывает данные, над которыми выполняются действия. Примером описания может служить предложение «В памяти следует отвести место для хранения целого числа, и это место мы будем обозначать А».

Исполняемые операторы для краткости часто называют просто операторами, а операторы описания – описаниями. Описания должны предшествовать операторам, в которых используются соответствующие данные. Операторы программы исполняются последовательно, один за другим, если явным образом не задан другой порядок.

# Б) Алфавит и лексемы.

Алфавит языка состоит из:

* Прописных и строчных букв латинского алфавита;
* Используется знак подчеркивания;
* Десятичных цифр ( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 );
* Шестнадцатеричных цифр ( 0,1, …, 9, А, В, …, F);
* Специальных символов;
* Составных символов.

# В) Константы.

Каждый элемент данных, используемый в программе, является константой или переменной.   
**Константами** называются элементы данных, значения которых в процессе выполнения программы не изменяются. Например, в языке Паскаль используются константы следующих типов:  
1) числовые – предназначены для представления числовых данных (целых и вещественных).  
2)символьные и строковые константы – отдельные символы и их последовательности. Заключаются в апострофы.

**Целые константы** делятся на десятичные и шестнадцатеричные (представляют собой знак $, непосредственно за которым следуют шестнадцатеричные цифры: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).

**Вещественные константы** записываются с точкой перед дробной частью. Либо целая, либо дробная часть может отсутствовать. Вещественная константа с порядком представляется в виде *мантиссы* и *порядка*. Мантисса записывается слева от знака **Е** или **е,** порядок справа от этого знака. Значение константы определяется как произведение мантиссы и возведенного в указанную в порядке степень числа 10. Пробелы внутри числа не допускаются.

**Символьные константы** служат для представления любого символа из набора, используемого в данном компьютере. Так как под каждый символ отводится 1 **байт** (единица хранения и обработки цифровой информации, **байт** считается равным восьми битам) , всего используется 256 символов, так как диапазон возможных значений 0-255.  
Десятичный код символа, обозначается предваряемым знаком **#**.А буква, предваряемым знаком **^**.

**Строковая константа** – это последовательность символов, расположенная на одной строке и заключается в апострофы. Если требуется представить сам апостроф, он дублируется. Всего 126 символов.

# Г) Имена, ключевые слова и знаки операций.

Имена в программах служат той же цели, что и имена людей, - чтобы обращаться к программным объектам и различать их, то есть идентифицировать. Поэтому имена также называют идентификаторами ( в информатике специальное наименование, имя элементарных данных, массивов данных, программ или других объектов, которые запрашиваются, обрабатываются и выдаются на выход ЭВМ).  
Имена дает программист, при этом следует соблюдать некоторые правила:

* Имя должно начинаться с буквы;
* Имя должно содержать только буквы, знак подчеркивания и цифры;
* Прописные и строчные буквы не различаются;
* Длина имени практически не ограничена (63 символа ).

Имена даются элементам программы, к которым требуется обращаться: переменным, константам, процедурам, функциям и т. д.).

**Ключевые слова** – это идентификаторы, имеющие специальное значение для *компилятора*. Их можно использовать только в том смысле, в котором они определены. Например, для описания переменных определено ключевое слово **var**. Имена, создаваемые программистом не должны совпадать с ключевыми словами.

Знак операции – это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются. Чаще всего знаки операции состоят из одного символа.   
Например:   
**1.**унарная операция **not**, унарный минус -,взятие адреса **@**.  
**2.**операция типа умножения: \*, /, div, mod, and.  
3.операция типа сложения: +, -, or.  
4.операции отношения: =, <, >,<=.

# Д) Переменные.

**Переменные,** в отличие от констант могут менять свои значения при выполнении программы. В программировании переменную можно трактовать как одну или несколько ячеек оперативной памяти компьютера (предназначена для временного хранения и передачи данных и команд процессору, для выполнения им операций), которым присвоено определенное. Содержимое этих ячеек может меняться, но имя остается неизменным. Каждое новое значение, записанное в ячейку памяти, «затирает» предыдущее значение, поэтому в любой момент времени переменная имеет только одно, текущее, значение.

В математике значение переменной в рамках определенной задачи неизменно. Именно поэтому высказывание, **а:=а+1** математик сочтет неверным. Тем не менее, для программиста это абсолютно правильная конструкция, которая задает вычисление суммы содержимого ячейки **а** и числовой константы **1** и занесение полученного результата в ту же ячейку **а**. после выполнения этого действия старое значение переменной **а** будет безвозвратно потеряно, так как одна ячейка памяти не может вместить сразу несколько значений. *Это очень важный момент в программировании.*

Все переменные, используемые в программе, должны быть описаны в разделе описания переменных, начинающемся со служебного слова **var**. Для каждой переменной задается ее имя и тип – integer (целое), real(реальное), char ( значение переменных в апострофах).

# Е) Пример программы.

Для того чтобы лучше представлять себе, о чем идет речь, рассмотрим простейшую программу на Паскале.

{1}  
  
{2}  
{3}  
{4}  
{5}  
{6}  
{7}

program ;   
var;  
 a, b, sum : integer;   
begin   
readln( a, b);   
sum := a + b;   
writeln( ‘Сумма чисел ‘, a, ‘и ‘, b, ‘ равна ‘, sum);   
end.

В первой строчке заглавие программы.

Во второй строке располагается оператор описания величин, которые будут использоваться в программе. Для каждой величины задается имя, по которому к ней будут обращаться, и ее тип. Словом VAR обозначается тот факт, что **a, b** и **sum –** переменные, то есть величины, которые во время работы программы могут менять свои значения. Для всех переменных задан целый тип, он обозначается **integer** (указания, что эти переменные целые числа). Тип необходим для того, чтобы переменным в памяти было отведено соответствующее место.

Исполняемые операторы программы располагаются между служебными словами **begin** и **end**, которые предназначены для объединения операторов и сами операторами не являются. Операторы отделяются друг от друга точкой с запятой.

Ввод с клавиатуры выполняется в четвертой строке с помощью стандартной процедуры с именем **readln.** В скобках после имени указывается, каким именно переменным будут присвоены эти значения. Для вывода результатов работы программы в шестой строке используется стандартная процедура **writeln**. В скобках через запятую перечисляется все, что мы хотим вывести на экран, при этом пояснительный текст заключается в апострофы. Например, если ввести в программу числа 2 и 3, результат будет выглядеть так:  
СУММА ЧИСЕЛ 2 И 3 РАВНА 5

В пятой строке выполняется вычисление суммы и присваивание ее значения переменной **sum.** Справа от знака *операции присваивания*, обозначаемой символами **:=**, находится так называемое выражение. **Выражение** – это правило вычисления значения. Выражения являются частью операторов.

Для того чтобы выполнить программу, требуется перевести ее на язык, понятный процессору,- в машинные коды. Этим занимается **компилятор**. Каждый оператор языка переводится в последовательность машинных команд, которая может быть весьма длинной. Компилятор планирует размещение данных в оперативной памяти в соответствии с операторами описания. Попутно он ищет синтаксические ошибки, то есть ошибки записи операторов.

После каждой строчки ставится точка с запятой ,кроме **begin** –после него ничего не ставится, а после **end** ставится точка.

# Заключение.

Новые языки и средства программирования появляются непрерывно, поэтому программист вынужден учиться всю жизнь. Следовательно, очень важно это делать быстро и эффективно. Для этого надо подходить к освоению каждого языка системно: выделить составные части, понять их организацию и взаимосвязь.

Моя работа основывалась на составных частях языка программирования и их назначения. Это является определенно «базой», которая лежит в основе программирования.

# Приложение №1(константы).



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Константы** | | | | | |
| **Целые** | | **Вещественные** | | **Символьные** | **Строковые** |
| **Десятичные** | **Шестнад-цатеричные** | **С плавающей точкой** | **С порядком** |  |  |
| 2 15 | $0101 $FFA4 | -0.26 .00521. | 1.2e4 0.1E-5 | ‘k’ #186 ^M | ‘абырвалг’ ‘I’’m fine’ |

Примечания : 1.2е4 значение 1.2\*104 ,  
 0.1Е-5 значение 0.1\*10-5 .

|  |  |  |  |
| --- | --- | --- | --- |
| and - и | export- экспорт | program - программа | var - переменные |
| begin - начать | file – файл  for - для | set - множество | xor - или |
| case - вариант | function - функции | then -тогда | of - из |
| const -константа | goto - перейти | to - увеличивая | or - или |
| div -деление нацело | if -если | type - тип | repeat - повторять |
| do - выполнять | In - в | until - до | for - для |
| downto –уменьшить до | nil-отсутствие | uses –включение библиотечных модулей | forward - опережающий |
| else - иначе | not - нет | with - с | inline - основной |
| end - конец | mod –остаток от деления | object - объект | packed -упакованный |

# Приложение №2 (ключевые слова).

Ключевые слова имеют специальный смысл, и переопределять их нельзя.

# Приложение №3( языки программирования).

|  |  |  |
| --- | --- | --- |
| Язык | Основное использование | Описание |
| Ада | В обороне | Высокого уровня |
| Ассемблер | Работы, требующие детального контроля за аппаратным обеспечением, быстрого исполнения и программ малого размера | Быстрый и эффективный, но требующий определенных усилий и навыков |
| Бейсик | В образовании, бизнесе, дома | Прост в изучении |
| С | Системное программирование, универсальное программирование | Быстрый и эффективный, широко используется как универсальный язык |
| С++ | В объектно-ориентированном программировании | Основан на языке С |
| Кобол | Программирование в бизнесе | Жестко ориентирован на коммерческие задачи, легко научиться, но очень много операторов |
| Форт | Управление приложениями | Использует инверсную польскую запись |
| Фортран | Научная работа и вычисления | Основан на математических формулах |
| Лисп | Искусственный интеллект | Язык символов с репутацией трудно изучаемого |
| Модула-2 | Системное программирование и программирование в режиме реального времени, универсальное программирование | Высоко структурирован, предназначен заменить Паскаль для приложений "реального мира" |
| Оберон | Универсальное программирование | Небольшой, компактный язык, соединяющий многие черты Паскаля и Модула-2 |
| Паскаль | Универсальный язык | Высоко структурирован |
| Пролог | Искусственный интеллект | Символьно-логическая система программирования, в начале предназначенная для решения теорем, но сейчас использующаяся чаще для решения задач, связанных с искусственным интеллектом |

# Приложение №4 (специальные символы).

|  |  |
| --- | --- |
| **Символ** | **Название** |
| + | Плюс |
| - | Минус |
| \* | Звездочка |
| / | Дробная черта |
| = | Равно |
| > | Больше |
| < | Меньше |
| [ ] | Квадратные скобки |
| ( ) | Круглые скобки |
| @ | Коммерческое а |
| { } | Фигурные скобки |
| . | Точка |
| , | Запятая |
| : | Двоеточие |
| ; | Точка с запятой |
| ‘ | Апостроф |
| # | Номер |
| $ | Знак денежной единицы |
|  | Пробел ( не имеет обозначения) |

# Приложение №5 ( составные символы).

|  |  |
| --- | --- |
| **Символ** | **Название** |
| := | Присваивание |
| < > | Не равно |
| . . | Диапазон значений |
| ( \* \* ) | Альтернатива { } |
| <= | Меньше или равно |
| >= | Больше или равно |
| ( . . ) | Альтернатива [ ] |

# Список литературы

1. Рапаков Г. Г., Ржеуцкая С. Ю.  
   Turbo Pascal для студентов и школьников. – СПб.: БХВ-Петербург, 2005. – 352 с.: ил.
2. Павловская Т. А.  
   Паскаль. Программирование на языке высокого уровня: Учебник для вузов – СПб.: Питер, 2006. – 393 с.: ил.
3. Консультант плюс