Лабораторная работа № 3

Выполнение программ в среде Free Pascal

Цель работы: ознакомиться со средствами, предоставляемыми ИС, для выполнения программ в системе **Free Pascal** и освоить их применение на примере простейших линейных программ. Закрепить навыки по подготовке и компиляции программ в ИС.

1. Теоретическая часть

1.1. Структура простейших линейных программ

Для выполнения работы требуется выполнить несколько простейших линейных программ. Попытаемся определить их структуру исходя из общих свойств, присущих программам, и цели данной лабораторной работы.

Структура программы на языке *Free Pascal* определяется следующим правилом грамматики:

<программа>*::=[<заголовок программы>] [<раздел описания используемых модулей>]<блок>*.

Из него следует, что блок и следующая за ним точка должны всегда присутствовать в программе. Заголовок программы является необязательным (т.к. заключен в скобки [] в правиле грамматики), но при проведении данной и последующих работ будет использоваться для задания в имени программы номера бригады и варианта задания. В разделе описаний в секции *USES* указываются имена используемых модулей. Так, например, для возможности выполнения в программе усовершенствованных операций (подпрограмм) ввода данных с клавиатуры и вывода их на экран требуется модуль *Crt*. Таким образом, легко убедиться, что все составляющие, указанные в правиле, присутствуют в программе (рис. 3.1).

Program имя программы; Необязательный заголовок

Uses Crt;
Const
Описание констант и инициализация переменных
Type
Описание типов, определяемых пользователем
Var
Описание переменных
Описание нестандартных

РАЗДЕЛ ОПИСАНИЙ

процедур и функций

Рис. 3.1. Структура простейшей программы

В структуре блока, определяемого правилом

<блок>::=<раздел описаний>BEGIN<раздел операторов>END,

можно выделить две части: раздел описаний и раздел операторов. Раздел описаний, в свою очередь, делится на ряд подразделов, а раздел операторов содержит операторы, разделенные символами ";". Содержимое этих двух частей блока определяется решаемой задачей. Однако можно выявить и некоторые общие закономерности.

Так, решение любой задачи предполагает получение некоторых результатов на основе конкретных исходных данных. Поэтому любая программа, после того как ее запустят на выполнение, должна в той или иной форме запрашивать исходные данные, проводить их обработку, а по окончании решения задачи выдавать полученные результаты. Действия по вводу исходных данных, их обработке и выводу результатов указываются в операторной части блока. Исходные данные могут быть введены в программу лишь как значения некоторых переменных. Переменные могут также использоваться и для сохранения значений результатов. Поэтому в программе должен быть раздел описания переменных, начинающийся со слова *Var*.

Подраздел (секция) констант в разделе описаний начинается с зарезервированного слова *Const* и требуется для инициализации (т.е. задания начальных значений) переменных. Подраздел (секция) нестандартных процедур и функций в разделе описаний используется для описания собственных подпрограмм (процедур и функций). Каждое описание подпрограммы начинается со слова *PROCEDURE* (для процедуры) или *FUNCTION* (для функции), за которым следуют имя и список формальных параметров, и заканчивается собственным блоком и последующей точкой с запятой.

1.2. Экраны и окна, поддерживаемые ИС

Обычно ввод исходных данных выполняется с клавиатуры в эхо-режиме, т.е. с отображением вводимых данных на экран дисплея. На этот же экран производится и вывод результатов. Система *Free Pascal* после ее вызова сначала снимает копию с экрана и только после этого выводит на него основное изображение ИС. В дальнейшем система поддерживает два состояния экрана, одно из которых соответствует работе в ИС, а другое - выполнению программы в текстовом окне (консольном окне).

Таким образом, имитируются два экрана. На первом экране отображаются меню и окна ИС, поэтому его можно назвать *экраном разработчика программы* (экран ИС). На другом окне видны подсказки и команды командной строки ОС, а также результаты выполнения операций по вводу-выводу данных, и его логично назвать *экраном пользователя*.

При выполнении программы автоматически поддерживается определенный порядок переходов между экранами.

Во-первых, сразу же после запуска программы восстанавливается экран пользователя и вся дальнейшая работа происходит только в нем. По завершении выполнения программы восстанавливается экран ИС.

Во-вторых, даже при выполнении программы *по шагам* переход в экран пользователя для ввода данных и при выводе результатов осуществляется автоматически, а по окончании выполнения этих операций также автоматически происходит возврат в ИС.

В случае небольших программ переход в экран пользователя и возврат в ИС происходит настолько быстро, что не удается как следует рассмотреть, какие результаты вывелись на экран. Можно задержать возврат в ИС путем имитации ввода данных перед самым завершением программы. Для этого перед последней строкой программы (*End* с точкой) необходимо поставить следующие операторы:

Write('Для возврата в ИС нажмите <*Enter*>');

Readln;

Чтобы из ИС снова вернуться в экран пользователя для просмотра полученных результатов, необходимо нажать *Alt+F5*, а для последующего возврата в ИС достаточно нажать *любую клавишу*.

Однако это становится неудобным при пошаговом выполнении программ, выводящих большие объемы данных на экран. В таком случае лучше окно Output держать открытым постоянно, для чего сделать следующее:

- размер окна редактирования уменьшить по вертикали и сместить это окно ниже, освободив вверху экрана ИС *Free Pascal* место под еще одно окно (окно *Output*);

- открыть окно *Output* (в меню *Debug* выбрать *Output*), уменьшить его размер и переместить выше окна редактирования на освободившееся место.

1.3. Режимы выполнения программ в ИС и курсор выполнения

ИС системы *Free Pascal* позволяет выполнять программы в двух режимах: *обычном* (автоматическом) и *режиме отладки.*

Под *обычным* будем понимать такой режим выполнения программы, когда программа, будучи запущена на выполнение, останавливается либо по достижении последнего *End* с точкой, либо по прерыванию, предусмотренному в самой программе. Этот режим имитируется системой **Free Pascal** путем восстановления экрана пользователя на все время выполнения программы. По окончании выполнения программы происходит возврат в ИС.

Режим отладки отличается тем, что прерывание выполнения программы осуществляется интегрированной средой, после чего также восстанавливается экран ИС. Та строка, на которой прерывается выполнение программы, отмечается яркой полоской, называемой далее *курсором выполнения* (не путать с обычным текстовым курсором). Последующее выполнение программы может быть продолжено только начиная со строки, на которую указывает этот курсор.

Если режим отладки сброшен (о чем свидетельствует *отсутствие* курсора выполнения в тексте программы), то независимо от режима, в котором запускается программа, она выполняется от начала.

Если режим отладки включен (о чем свидетельствует *наличие* курсора выполнения в тексте программы), то программа выполняется со строки, на которую указывает курсор выполнения. Чтобы после частичного выполнения программы в режиме отладки перезапустить ее от начала, необходимо сбросить этот режим (нажать Ctrl+F2).

Если программа еще не откомпилирована или после компиляции в нее были внесены изменения, то при попытке ее выполнения программа сначала откомпилируется и только затем запустится на выполнение.

1.4. Команды ИС для выполнения программ

Команды ИС по выполнению программ находятся в меню *Run* основного меню. Меню *Run* состоит из восьми команд и имеет следующий вид:

Run (*Ctrl+F9*) - запуск программы на исполнение в обычном (автоматическом) режиме. Требуемые для запуска параметры указываются с помощью команды *Parameters...*, находящейся в этом же меню.

Step over (*F8*) - пошаговое выполнение программы. Вызовы процедур и функций выполняются как один оператор (как один шаг).

Trace into *(F7)* - пошаговое выполнение программы. При вызове процедуры или функции происходит вход в ее текст и пошаговое выполнение ее операторов.

Go to cursor (*F4*) - выполнение участка программы от текущей строки пошагового выполнения программы до строки, в которой находится курсор.

Until return *(Alt+F4)* - выполняет текущую процедуру до точки выхода из нее.

Run Directory… - открывает диалоговое окно *Change Directory*, в котором должна быть указана папка с файлом программы.

Parameters... - открывает диалоговое окно, в котором указываются параметры для выполняемой программы.

Program reset (*Ctrl+F2*) - завершает сеанс отладки программы и освобождает занимаемую ею память.

Можно заметить, что все пункты меню *Run* продублированы «горячими клавишами», поскольку выполнение таких команд через меню неудобно.

Режим отладки представлен в меню *Run* тремя командами.

Команда *Go to cursor* (*F4*) вызывает выполнение программы в обычном режиме до строки, на которую установлен текстовой курсор. При этом строка, с которой начинается выполнение, определяется по описанным в п.1.3 правилам.

Команды *Trace into* (*F7*) и *Step over* (*F8*) представляют собой так называемые команды трассировки, вызывающие *построчное* выполнение программы. При сброшенном режиме отладки любая из этих команд подготавливает очередной сеанс отладки. При этом инициализируются переменные, описанные в подразделе раздела описаний, после чего курсор выполнения устанавливается на первый *Begin* программы. Чтобы перевести курсор выполнения на следующую строку, требуется еще одно нажатие *F7* или *F8*, как бы заставляющее выполниться *Begin*, что может вызвать удивление, если не знать, что за *Begin* действительно скрывается часть кода (т.е. команд ЭВМ), называемого инициализирующим (т.е. выполняющим начальные установки) кодом. Перечисленные действия выполняются и по двум предыдущим командам, когда программа запускается от начала, однако они не так заметны, поскольку не происходит остановки на строке с первым *Begin* программы.

Нечто подобное наблюдается при достижении последнего *End* программы, для прохождения которого, как и выше, требуется отдельное нажатие *F7* или *F8*. Однако в отличие от *Begin End* соответствует не инициализирующему коду, а завершающему, т. е. обеспечивающему выход из программы.

В остальном различие между командами *Trace into* (*F7*) и *Step over* (*F8*) становится заметным только при появлении в очередной выполняемой строке программы обращения к подпрограмме (т.е. к процедуре или функции). По команде *Trace into* (*F7*) процедуры или функции также выполняются по строкам. При достижении конца процедуры осуществляется

возврат в основную программу. Однако трассировка подпрограмм стандартных модулей не осуществляется.

Команда *Step over* (*F8*) похожа по своим действиям на *Trace Into.* Отличие состоит в том, что не происходит “заходов” в процедуры и функции, они выполняются как один оператор основной программы.

Команда *Until return (Alt+F4)* позволяет выполнить текущую подпрограмму до точки выхода из нее.

Оставшиеся команды меню *Run* не вызывают выполнения программы.

Команда *Program reset* (*Ctrl+F2*) выполняет сброс режима отладки и подготовку к выполнению программы от начала. Внешне это выражается в гашении курсора выполнения.

Команда *Parameters* позволяет осуществить запуск программы с соответствующими параметрами

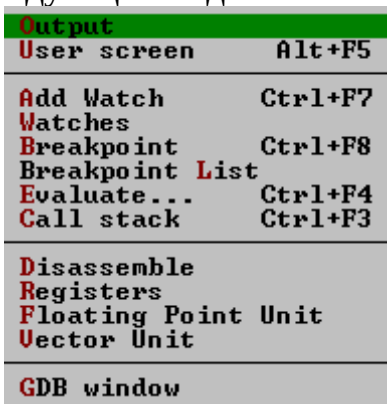
Команда *User* screen *(Alt+F5)* находится в пункте *Debug* основного меню и позволяет из любого окна ИС перейти в экран пользователя.

1.5. Дополнительные возможности по прерыванию выполнения программы

Рассмотренные выше команды *Trace into*, *Step over* и *Go to cursor* позволяют осуществлять контроль правильности выполнения программы с любой степенью детальности. Однако когда строки, в которых необходимо контролировать правильность выполнения программы, заранее известны, а программа запускается многократно на различных наборах исходных данных, применение перечисленных команд становится неэффективным. Действительно, вместо того, чтобы каждый раз подводить текстовой курсор к одним и тем же строкам, до которых требуется выполнять программу, гораздо удобнее один раз некоторым образом пометить такие строки и возложить работу по остановкам программы в этих строках на ИС. Работа в таком режиме называется отладкой с использованием точек прерывания (*BreakPoints*) или *точек останова*.

1.5.1. Использование точек останова

ИС *Free Pascal* предоставляет необходимые средства по установке, удалению и просмотру точек останова. Подменю точек останова *Breakpoints* находится в пункте *Debug* основного меню. Меню *Debug* состоит из тринадцати команд и имеет следующий вид:



Output - открывает окно *Output*, в котором отображаются экран ОС и результаты работы программы (в режиме командной строки, т.е. исключая графику).

User screen - просмотр результатов работы программы, включая графику, в полноэкранном режиме.

Add watch - открывает диалоговое окно *Add Watch*, в котором программист может указать выражение или имя переменной, значение которых его интересует при выполнении отладки.

Watches - открывает окно *Watch*, в котором пользователь может вывести для себя информацию о значениях переменных и выражений программы, которые требуются ему при отладке.

Breakpoint - точки в тексте программы (точки останова), в которых будет приостанавливаться ее работа для выполнения отладочных действий.

Breakpoint List - открывает диалоговое окно *Breakpoint List*, с помощью команд которого можно управлять условными и безусловными точками прерывания (*breakpoints*).

Evaluate... - открывает окно *Evaluate and modify*, в котором можно указать выражение, значение которого требуется определить, просмотреть значения переменных и элементов данных программы и изменить их.

Call stack - открывает стек вызовов, т.е. список (последовательность) адресов подпрограмм программы, вызванных до подпрограммы, выполняющейся в данный момент.

Disassemble - показывает стек вызовов.

Registers - открывает окно *Registers*, содержащее информацию о текущем содержимом регистров процессора.

Floating Point Unit - открывает окно, содержащее информацию о текущем содержимом регистров сопроцессора.

Vector Unit - открывает окно, содержащее информацию о текущем содержимом регистров MMX.

GDB Window - показывает консоль отладчика GDB.

Для пометки строки программы как точки останова достаточно подвести к ней текстовой курсор и нажать клавиши *Ctrl+F8*. Все символы строки будут выделяться более ярко на общем фоне текста программы.

В программе может быть столько точек останова, сколько нужно для отладки программы. Они не исчезают по завершении выполнения программы. Чтобы снять пометку со строки программы (т.е. удалить точку останова), необходимо подвести курсор к этой строке и нажать клавиши *Ctrl+F8*. Строка программы "погаснет" (сольется с общим фоном текста программы). Кроме того, имеется возможность одной командой удалить из программы все точки останова.

Для выполнения вышеперечисленных действий следует войти в пункт *Debug/Breakpoints* и нажать *Enter*. Раскрывается новое окно, в котором можно выбрать команды, относящиеся к точкам останова: *Edit* (редактирование), *Delete* (удаление), *View* (просмотр), *Clear all* (удаление всех).

Многократное повторение команды *View* позволяет обойти все точки останова в том порядке, в котором они устанавливались.

1.5.2. Использование клавиш *Ctrl+Break*

Все команды прерывания выполнения программы, рассмотренные выше, используются, когда известно, на какой строке или строках произойдет прерывание. Однако в ряде случаев возникает необходимость в незапланированном прерывании выполнения программы, например, если она начала заведомо неправильно работать: выдает неправильные результаты, зациклилась и т.д.

Выполнение программы можно прервать в любой момент времени, нажав клавиши *Ctrl+Break*. Она остановится в том месте, где ее застало нажатие этих клавиш, а ее текст появится в окне редактирования. При этом ИС выдаст сообщение, приведенное на рис. 3.2, которое можно перевести как: “Прерывание пользователем программы *LAB3.PAS*. Нажмите клавишу *Esc*”. Если в ответ нажать клавишу *Esc*, то сообщение исчезает и появляется курсор выполнения, указывающий на строку, с которой можно продолжить выполнение программы. Наличие курсора выполнения говорит о том, что установлен режим отладки, т.е. выполнение программы можно только продолжить со строки, на которой оно прервалось. Поэтому для перезапуска программы необходимо сначала сбросить режим отладки, нажав клавиши *Ctrl+F2*.

User break in LAB3.PAS. Press Esc.

Рис. 3.2. Реакция ИС на прерывание выполнения программы

Однократное нажатие клавиш *Ctrl+Break* может сразу и не привести к прерыванию программы. Это вызвано тем, что отлаживаемая программа выполняется под управлением ОС, которая время от времени обращается к Базовой системе ввода/вывода (*BIOS).* Поэтому машинная команда, выполняемая в некоторый момент времени, может относиться к одной из перечисленных программ. Система *Free Pascal* “знает” об этом и, обнаружив нажатие клавиш *Ctrl+Break*, ожидает перехода на команду, с которой начинается следующая строка исходного текста программы. Если в этот момент программа ожидает ввода данных с клавиатуры, например по оператору *Readln*, то до завершения их набора она будет продолжать “висеть”, так и не реагируя на первое нажатие клавиш *Ctrl+Break.*

В таком случае можно дать завершиться начатому выполнению текущей строки программы и получить сообщение, приведенное на рис. 3.2, или повторно нажать клавиши *Ctrl+Break*, прервав процесс ожидания начала выполнения следующей строки исходного текста. Во втором случае на экран выдается сообщение, приведенное на рис. 3.3 и переводимое как “*Выполнение программы остановлено по прерыванию пользователя. Нажмите клавишу Esc*”.

User break, program terminated. Press Esc.

Рис. 3.3. Реакция ИС на двойное нажатие клавиш *Ctrl+Break*

Однако после нажатия клавиши *Esc* курсор выполнения не появляется, поскольку такое прерывание программы является аварийным. Дальнейшее выполнение программы возможно только с ее начала.

1.6. Правила ввода исходных данных

При вводе исходных данных в программу необходимо учитывать следующее:

- ввод числа завершается только по нажатии клавиши *Enter*;
- пока не нажата клавиша *Enter*, неправильно набранную часть числа можно стереть, нажав нужное число раз клавишу “*BackSpace*”;
- после нажатия клавиши *Enter* неправильно введенное число исправить невозможно.

Простейшие программы, используемые в лабораторной работе, не защищены от ошибок ввода исходных данных, поэтому в случае неисправимого неправильного ввода данных необходимо, если требуется, прервать выполнение программы, нажав клавиши *Ctrl+Break*, а затем перезапустить программу и повторить ввод.

Ошибки ввода данных могут быть *синтаксические* и *семантические*.

Синтаксические ошибки возникают при нарушении синтаксических правил записи данных, например в результате нажатия на клавишу с буквой, а не с цифрой, при вводе целого числа.

При выполнении программы синтаксическая ошибка ввода числа вызывает прерывание выполнения программы и выдачу на экран системного сообщения об ошибке вида: “*Runtime error at 20E8:0024”*, что переводится как “*Ошибка времени выполнения 106 по адресу 20E8:0024*”.

Если программа запущена из ИС, то синтаксическая ошибка ввода числа, кроме выдачи только что описанного сообщения на экран пользователя, вызывает переход в окно *Edit*, в верхней строке которого появляется объяснение ошибки в следующем виде: “*Error 106: Invalid numeric format”*, которое переводится как: “*Ошибка 106: Неправильный* числовой формат”. Это сообщение исчезает после нажатия любой клавиши.

Семантические (т.е. смысловые) ошибки в общем случае могут привести к недопустимости выполнения некоторых операций с данными, например таких, как деление, однако в этой лабораторной работе они не приведут к выдаче каких-либо сообщений.

2. Практическая часть

2.1. Порядок выполнения работы

1. Вызвать ИС *Free Pascal*.
2. Получить у преподавателя файл *LAB3.PAS* и загрузить его в окно *Edit*. Использовать любой из известных способов загрузки и создания файлов, изученных в лабораторной работе № 1.
3. Откомпилировать программу (пункт *Compile/Compile)*. Убедиться в отсутствии ошибок, в противном случае исправить их.
4. Окончательно откомпилировать программу (пункт *Compile/Build)* и убедиться, что на диске создан файл *LAB3.EXE* с программой, готовой к запуску на выполнение.
5. Завершить работу с ИС, используя команду *File/Quit (Alt-X).*
6. Найти на диске и выполнить созданную компилятором программу lab3.exe.
7. По запросу программы ввести исходные данные. Исходные данные взять в соответствии с вариантом задания (по указанию преподавателя) из таблицы 1 «Варианты заданий исходных данных» документа «Варианты заданий для ЛР3.doc». При вводе исходных данных надо учитывать правила ввода, приведенные в п.1.6.
8. При выполнении программы наблюдать, какое окно открылось, что в нем отображается и что с этим окном можно сделать (уменьшить, увеличить, переключиться). Результаты наблюдения записать в отчет вместе с результатами выполнения программы.
9. Снова войти в ИС *Free Pascal* (и, если окно редактирования пустое, снова загрузить в ИС файл *LAB3.PAS)*. Настроить окно Output в соответствии с рекомендациями, приведенными в конце п.1.2.
10. Выполнить программу с помощью команды *Run/Run* (*Ctrl+F9*). По запросу программы ввести исходные данные. Исходные данные взять в соответствии с вариантом задания (по указанию преподавателя) из таблицы 1 «Варианты заданий исходных данных» документа «Варианты заданий для ЛР3.doc». При вводе исходных данных надо учитывать правила ввода, приведенные в п.1.6.
11. При выполнении программы наблюдать, какое окно открылось и что с ним можно сделать (уменьшить, увеличить, переключиться). Результаты наблюдения записать в отчет вместе с результатами выполнения программы.
12. Выполнить программу, используя команду *Run/Trace into* (*F7*). Занести (из окна редактирования) в отчет последовательность номеров операторов выполнения программы и изменения в окне Output после каждого нажатия клавиши *F7*.
13. Выполнить программу, используя команду *Run/Step over* (*F8*). Занести (из окна редактирования) в отчет последовательность номеров операторов выполнения программы и изменения в окне Output после каждого нажатия клавиши *F8*.
14. Выполнить программу по команде *Run/Go to cursor* (*F4*). Номер строки программы выбрать в соответствии с вариантом задания из таблицы 2 «Варианты заданий номеров строк» документа «Варианты заданий для ЛР3.doc». Продолжить выполнение программы в пошаговом режиме (*F7*).

Занести в отчет последовательность номеров операторов выполнения программы и изменения в окне Output после каждого нажатия клавиши *F7*.

15. Поставить точку останова в заданной строке программы. Номер строки программы выбрать в соответствии с вариантом задания из таблицы 3 «Варианты расстановки точек останова». Выполнить программу до точки останова в обычном режиме, после точки останова - в пошаговом. Занести в отчет последовательность номеров операторов выполнения программы и изменения в окне Output после каждого нажатия клавиши *F7/F8*. Удалить поставленную точку останова.

16. Используя клавиши (*Ctrl-F8*), создать несколько (2 - 3) дополнительных точек останова (в соответствии с вариантом задания из таблицы 3 «Варианты расстановки точек останова» документа «Варианты заданий для ЛР3.doc»). Проделать для этих дополнительных точек останова все те действия, которые выполнялись в п.15.

17. Просмотреть точки останова, проставленные в предыдущем пункте. Записать в отчет номера строк с точками останова в том порядке, в котором они были показаны ИС.

2.2. Требования к содержанию отчета

Отчет о лабораторной работе должен включать:

1. Конспект теоретической части.
2. Текст исследуемой программы.
3. Последовательности номеров операторов выполнения программы для разных режимов выполнения и отладки программы.

2.3. Контрольные вопросы

1. Какую структуру имеет простейшая линейная программа ?
2. Дайте определение понятия «блок программы».
3. Какие экраны поддерживает система *Free Pascal*?
4. Какой порядок перехода между экранами поддерживается при выполнении программы: а) в обычном режиме; б) в режиме отладки?
5. Какими командами осуществляется переход между экраном пользователя и экраном ИС?
6. Какие окна есть в ИС?
7. В каких режимах возможно выполнение программы?
8. Что такое обычный режим выполнения?
9. Что такое курсор выполнения?
10. Как реализуется режим отладки?
11. Как сбросить режим отладки?
12. Для чего используется команда *Go to cursor*?
13. В чем различие между двумя командами трассировки?
14. Что такое точка останова? Как ее установить в программе?

15. Как удалить точку останова? Что при этом удаляется из текста программы?

16. Как выборочно удалить точки останова?

17. С помощью какой команды можно просмотреть все точки останова?

18. В каком порядке показываются точки останова?

19. Как можно прервать выполнение программы?

20. Почему для прерывания программы может оказаться недостаточно однократного нажатия Ctrl+Break?

21. Что происходит при двойном нажатии Ctrl+Break?

22. Какие действия следует осуществить при неправильном вводе данных, если программа запущена: а) из ОС; б) из ИС?

23. Какие ошибки возможны при вводе данных?