DR. BOB DAVIDOV

Знакомство с компилятором GNAT языка Ада для программирования систем управления автоматизированными комплексами

Цель работы: Получить представление об интегрированной среде разработки программ GNAT Programming Studio для специализированных платформ.

Задача работы: На примере готовой программы на языке Ада познакомиться со средствами разработки программ GNAT Programming Studio.

Приборы и принадлежности:.Персональный компьютер, компилятор GNAT.

ОБЩИЕ СВЕДЕНИЯ

Язык программирования Ада (Ada) был создан для обеспечения надежности программного кода систем управления автоматизированными комплексами, прежде всего, бортовыми системами управления военными объектами (кораблями, самолётами, танками, ракетами, снарядами и т.п.) работающими в реальном времени. По утверждению сторонников этого языка программирования, разработка программного обеспечения на Аде в целом обходится на 60 % дешевле, а разработанная программа имеет в 9 раз меньше дефектов, чем при использовании языка Си.

GNAT Programming Studio (GPS) – открытая (бесплатная) многоязычная интегрированная среда разработки программ включающая редактор, компилятор, отладчик и профилировщик. Кроме языка Ада GPS поддерживает языки C, JavaScript, Pascal и Python. GPS использует компиляторы из коллекции компиляторов GNU.

НАСТРОЙКА GNAT НА КОМПИЛЯТОР ТРЕБУЕМОГО ТИПА

Изначально среда GNAT с компилятором GCC настроена на создание исполняемого кода для x86 контроллера. Для подключения к среде GNAT компилятора GCC, настроенного на создание кода для другого контроллера (например, хранимого в папке C:\ARM Toolchain), необходимо отредактировать системную переменную окружения Path, как показано на Рис. 2.

Рис. 1. Структура каталогов компилятора GCC.

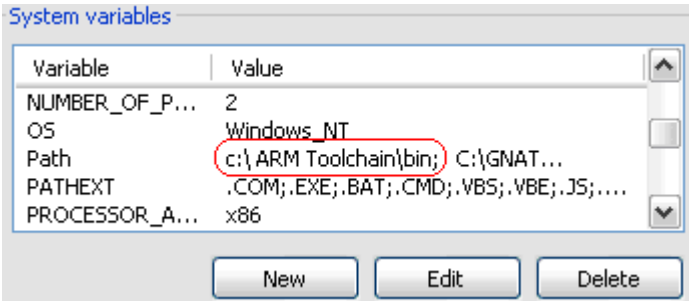
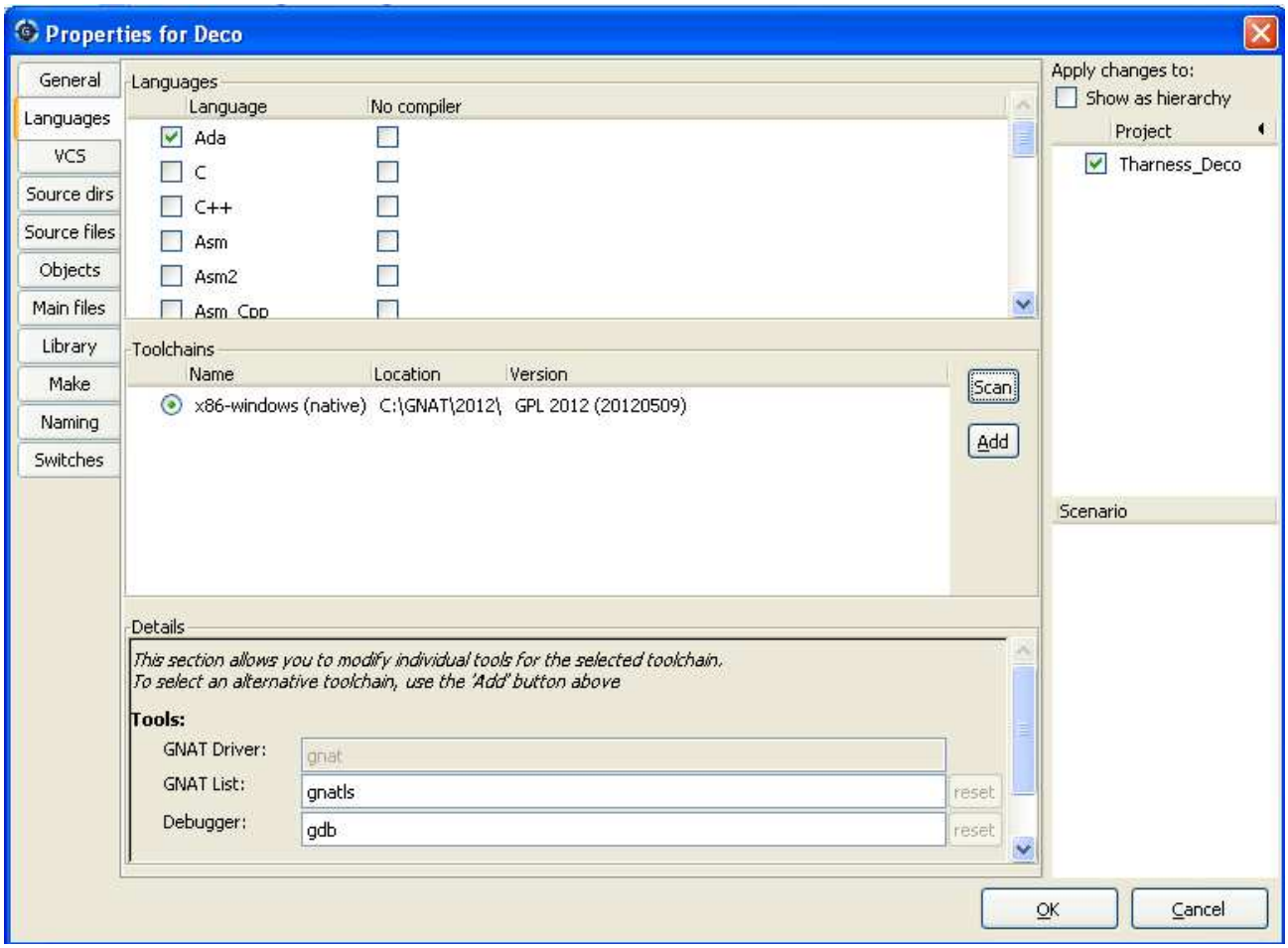
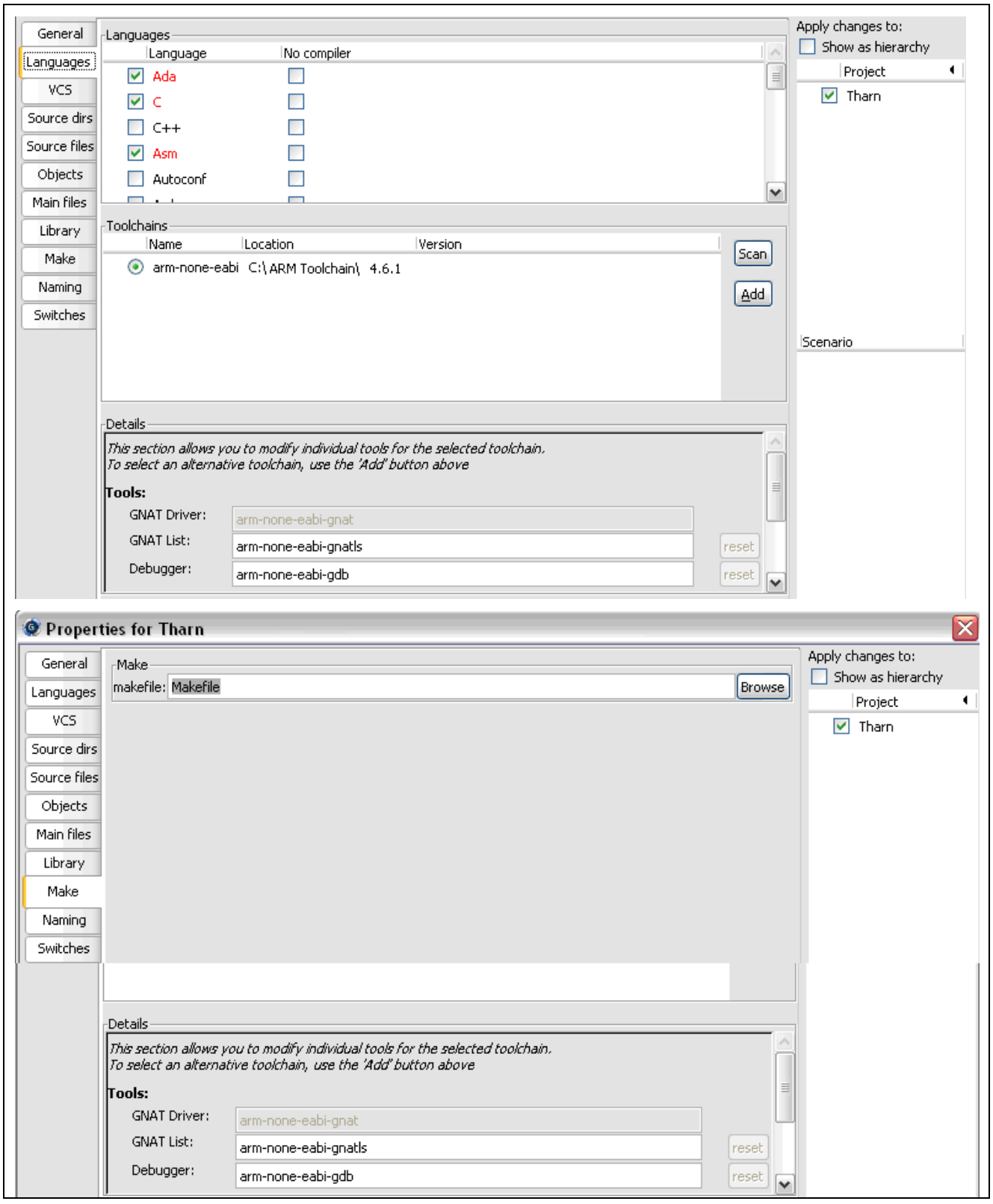
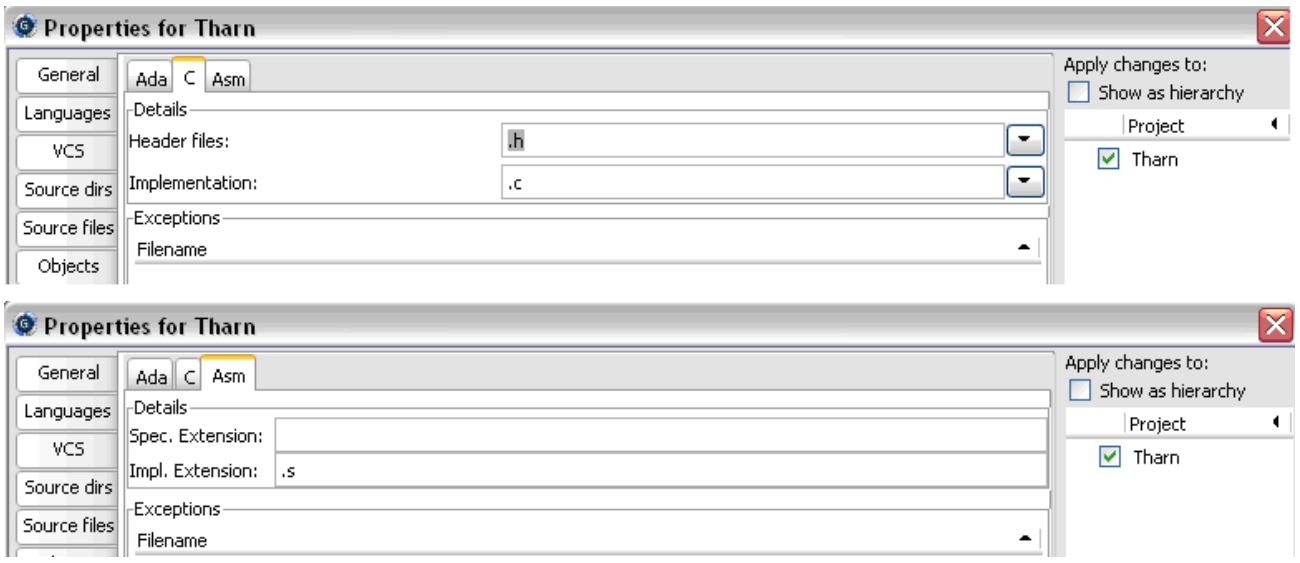


Рис. 2. Редактирование переменой окружения Path. My Computer > Properties > Advanced > Environment Variables > System variables > Path > Edit >



Исходное состояние. Компилятор **x86** процессора.


General
Languages
VCS
Source dirs
Source files
Objects
Main files
Library
Make
Naming
Switches
Languages
Language
No compiler
Ada
C
C++
Asm
Autoconf
Toolchains
Name
Location
Version
arm-none-eabi
C:\ARM Toolchain\
4.6.1
Scan
Add
Details
This section allows you to modify individual tools for the selected toolchain.
To select an alternative toolchain, use the 'Add' button above
Tools:
GNAT Driver:
arm-none-eabi-gnat
GNAT List:
arm-none-eabi-gnatls
Debugger:
arm-none-eabi-gdb
reset
reset
Apply changes to:
Show as hierarchy
Project
Tharn
Scenario
Properties for Tharn
General
Languages
VCS
Source dirs
Source files
Objects
Main files
Library
Make
Naming
Switches
Make
makefile:
Makefile
Browse
Details
This section allows you to modify individual tools for the selected toolchain.
To select an alternative toolchain, use the 'Add' button above
Tools:
GNAT Driver:
arm-none-eabi-gnat
GNAT List:
arm-none-eabi-gnatls
Debugger:
arm-none-eabi-gdb
reset
reset
Apply changes to:
Show as hierarchy
Project
Tharn



Подключение нового компилятора и языков программирования.

Рис. 3. Подключение нового компилятора и языков программирования в среде GNAT: меню > Project > Edit Project Properties.

Примечание: Настройка среды GNAT может выполняться и через Makefile.

НАСТРОЙКА КОМПИЛЯТОРА ПРОГРАММЫ

В соответствии с настройками компилятор помогает найти ошибки разного типа.

Настройка компилятора может выполняться через главное меню: Main Menu > Project > Edit Project Properties или с помощью gpr файла.

Для начальной настройки компилятора в меню среды GNAT предлагаются следующие установки.

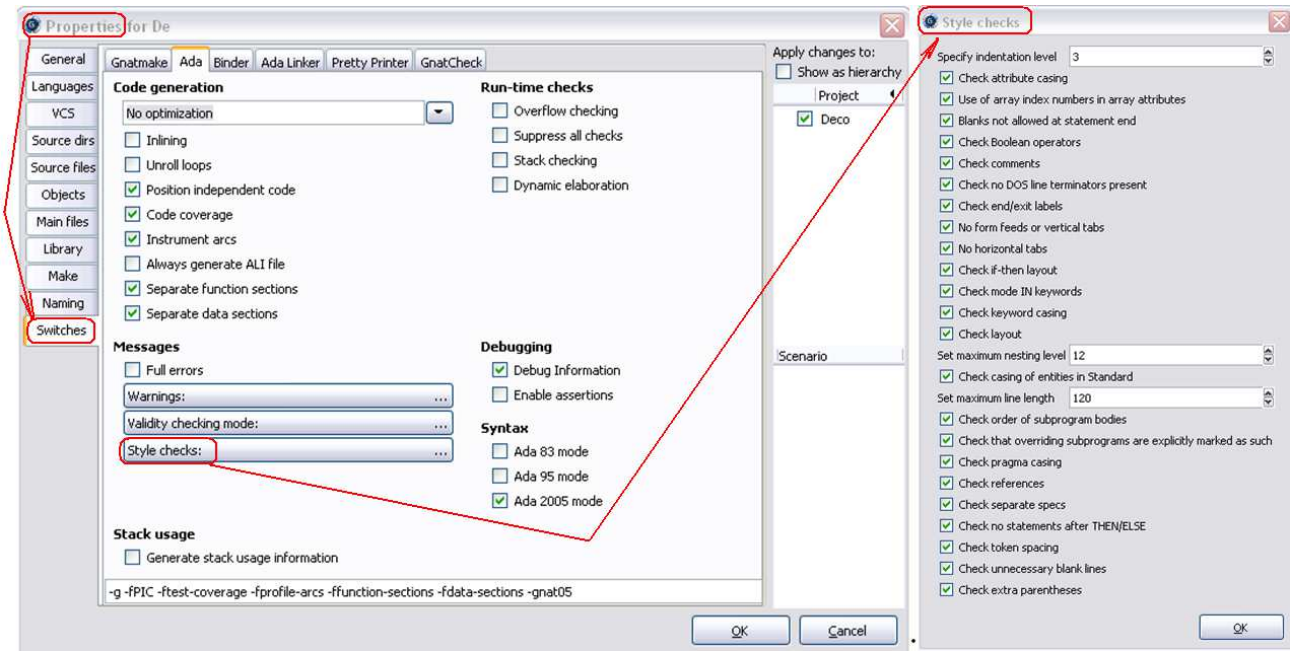


Рис. 4. Параметры настройки компилятора

Компилятор выдает список ошибок при их наличии. Для проверки нахождения ошибок в коде программы целесообразно использовать компиляцию отдельного модуля которая вызывается командой меню > Build > Compile file ( ).

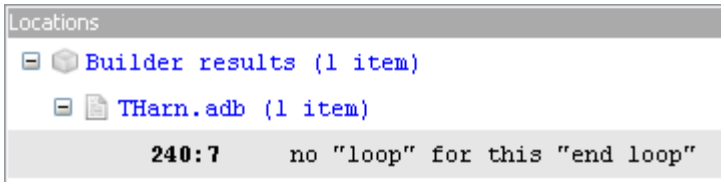


Рис. 5. Пример вывода сообщения об ошибке в окне Location.

```
project Tharn is

   for Object_Dir use "obj";
   for Exec_Dir use ".";
   for Main use ("Tharn.adb");
   for Source_Dirs use (".", "..\..\Alg");

   package Builder is
      for Default_Switches ("ada") use ("-g");
   end Builder;

   package Compiler is
      for Default_Switches ("ada") use ("-g", "-gnat05");
   end Compiler;
```

```
   package Linker is
      for Default_Switches ("ada") use ("-g");
   end Linker;


end Tharn;
```

Рис. 6. Пример настройки компилятора и редактора связей (Compile, Linker, Builder) в файле gpr.

КОМПИЛЯЦИЯ КОДА В EXE ФАЙЛ

Компиляция программного кода в исполняемый exe файл вызывается командой меню > Build > Project > Build All ().

Примечание: Чтобы удалить файлы предыдущей компиляции используйте команду меню > Build > Clean > "Clean All" ()

ОТЛАДЧИК ПРОГРАММЫ

Загрузка отладчика Debugger выполняется после создания exe файла по команде меню > Debug > Initialize > exe файл

Чтобы установить / удалить точку остановки выполнения программы (breakpoint) щелкните по точке  второй колонки соответствующей линии, показанной на Рис. 7. Красная точка отображает место, где остановится запущенная на выполнение программа.

Примечание: Для работы с точками останова необходимо установить следующие параметры отладчика.

- Меню > Project > Edit Project Properties > Switches > Gnatmake > Compilation > Debug information
- Меню > Project > Edit Project Properties > Switches > Ada > Debugging > Debug Information
- Меню > Project > Edit Project Properties > Switches > Ada Linker > Debug information

Чтобы увидеть значение программной переменной в точке останова укажите на нее мышью, нажмите правую кнопку и выберите Debug > Display. Имя и значение вычисленной переменной появятся в окне отладчика “Debugger Data” (см. Рис. 7).

Для копирования значения переменной укажите на нее в программе мышью, нажмите правую кнопку, выберите Debug > Print, выделите отображаемое значение и скопируйте его в буфер Paste (Ctrl+C).

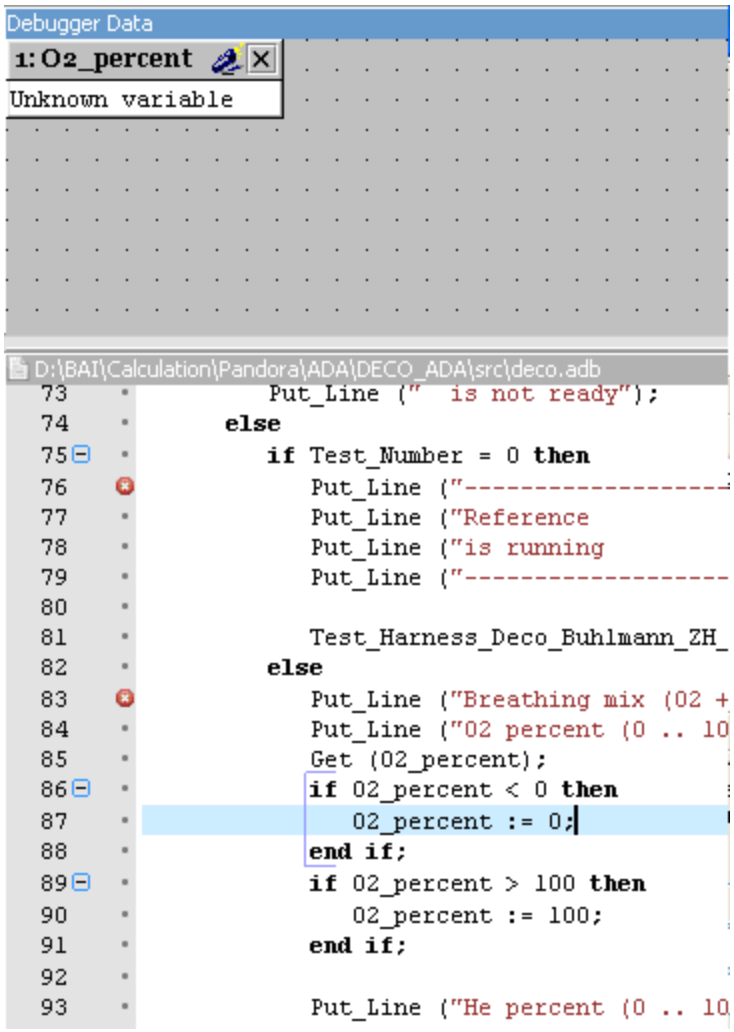


Рис. 7. Редактирование переменой окружения Path. My Computer > Properties > Advanced > Environment

Для выполнения команд программы по отладчиком **Debugger** используйте следующие GPS команды

- Меню > Debug > Run/Continue или <F2>/<F2> или  начальный / повторный запуск
- Меню > Debug > Step или <F5> или  пошаговое выполнение
- Меню > Debug > Next или <F6> или  пошаговое выполнение
-  выполнение до следующей breakpoint или выхода из модуля

Для выхода из отладчика выполните меню > Debug > Terminate

ЗАПУСК И ОСТАНОВ EXE ФАЙЛА В GNAT

Для запуска исполняемого файла в среде GNAT выполните команду меню > Build > Run > “имя файла”

или нажмите выделенную на рисунке кнопку: 

Ввод и вывод данных исполняемого файла и результаты отображается в следующем окне Run среды GNAT

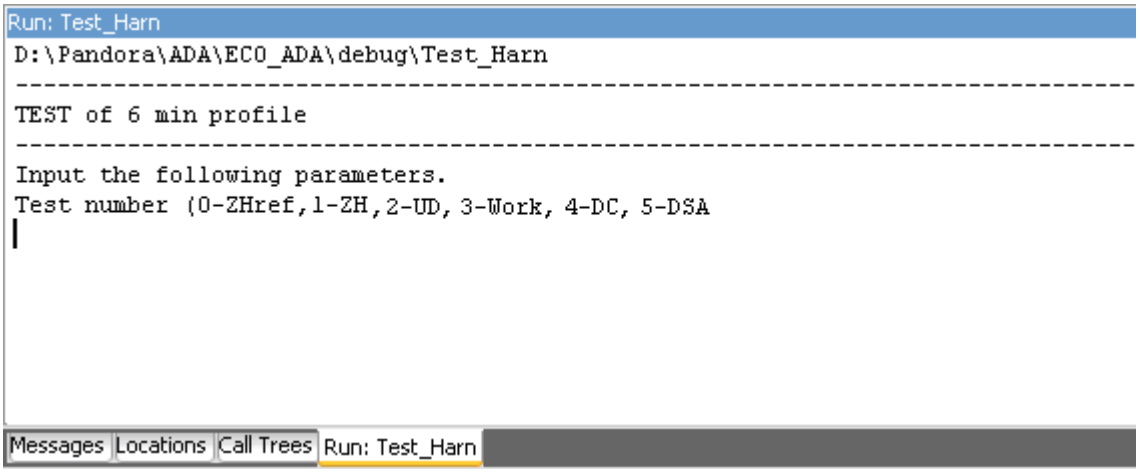


Рис. 8. Окно Run ввода-вывода исполняемого файла в среде GNAT

Остановку работающего исполняемого (**exe**) файла в GNAT можно выполнить командой

меню > Tools > Interrupt или Shift + Ctrl + C

или при помощи менеджера программ операционной среды Windows, выполнив последовательность:

Ctrl + Alt + Del > Windows Task Manager > Processes > выбор exe файла в списке "Image Name" > End Process

ЗАПУСК ФАЙЛА ЧЕРЕЗ КОМАНДНОЕ ОКНО

Если программа, скомпилированная в среде GNAT, не содержит команд ввода (например, Get), то при запуске exe файла в среде Windows, зачастую, не удается увидеть результаты программы поскольку окно с результатами открывается на мгновение время выполнения программы.

В случае, если исполняемый файл необходимо загружать за пределами среды GNAT и при этом необходимо сохранять на дисплее окно с выводимыми результатами выполните запуск файла через командную строку Windows:

Start 

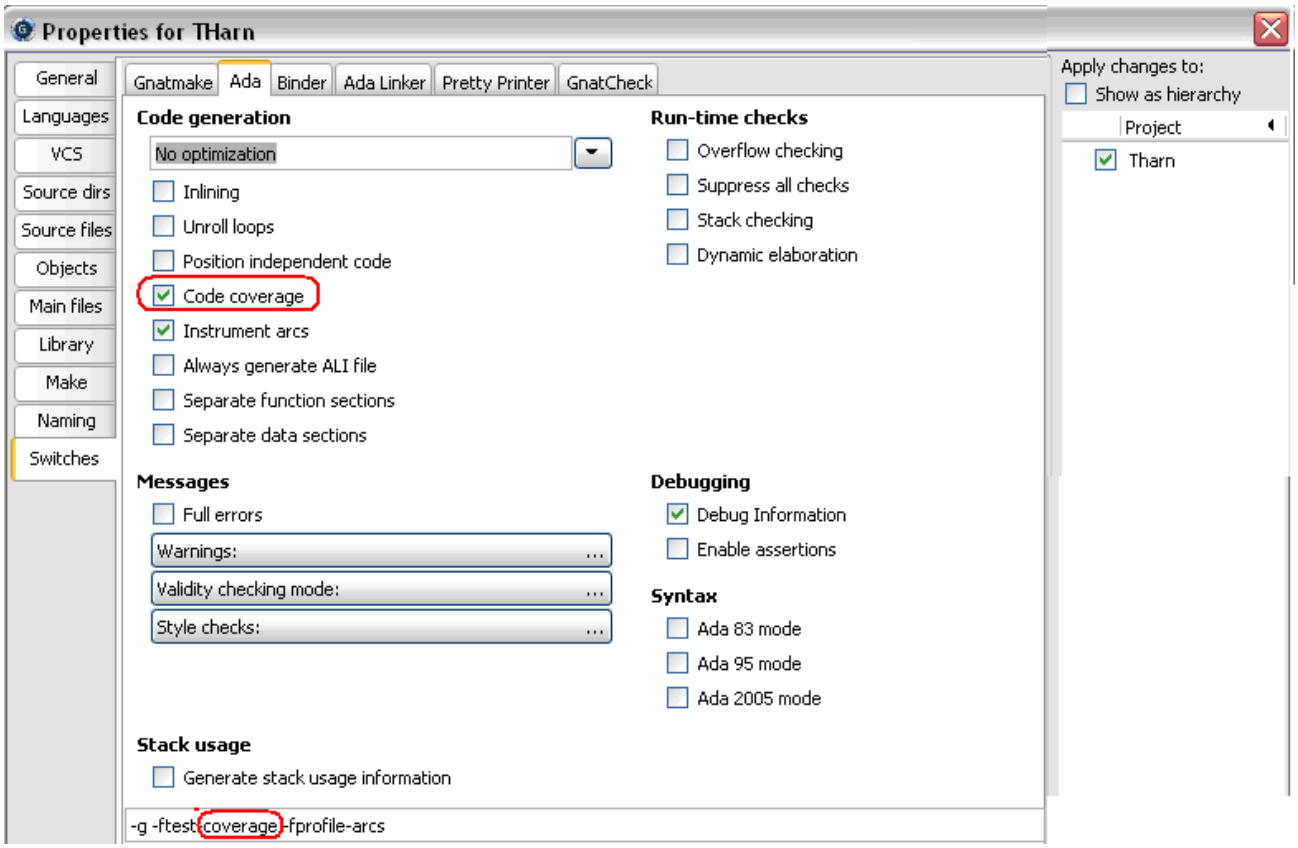
> Run > Open > cmd

В открывшемся окне, используя команду cd (Change Directory - смена каталога), настройте путь на директорию с файлом, введите имя файла и нажмите клавишу Enter.

Примечание: Можно использовать копирование имени через буфер paste и ввод имени в командную строку командой paste, которая вызывается нажатием на правую клавиши мыши в командной строке.

ТЕСТИРОВАНИЕ ПРОГРАММ В GNAT

Модуль **gcov**, запускаемый из среды GNAT, анализирует выполнение программы на прохождение выбранных тестов. Модуль помогает выявить те части программы, которые не участвовали в тестировании. Информация о таких частях программы помогает оптимизировать тест программы и вектор входных параметров.



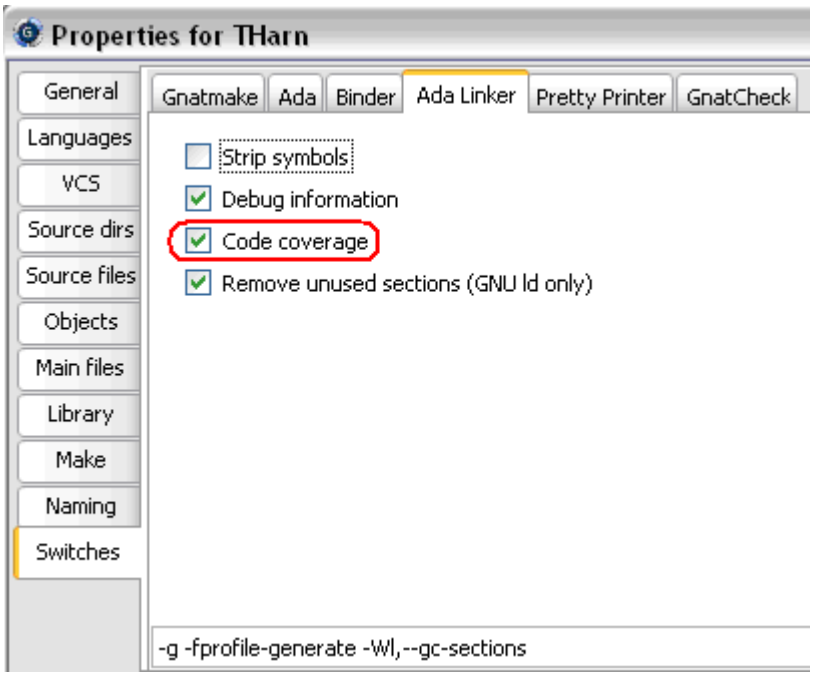


Рис. 9. Настойка среды GNAT на тестирование полноты выполнения кода (**Test Coverage**).

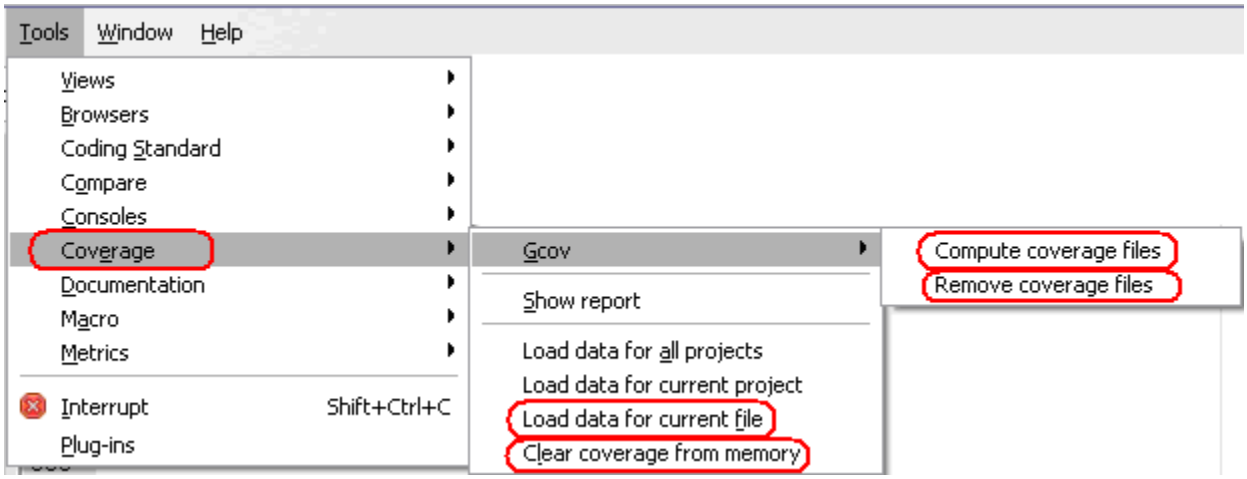


Рис. 10. Команды выполнения теста Coverage.

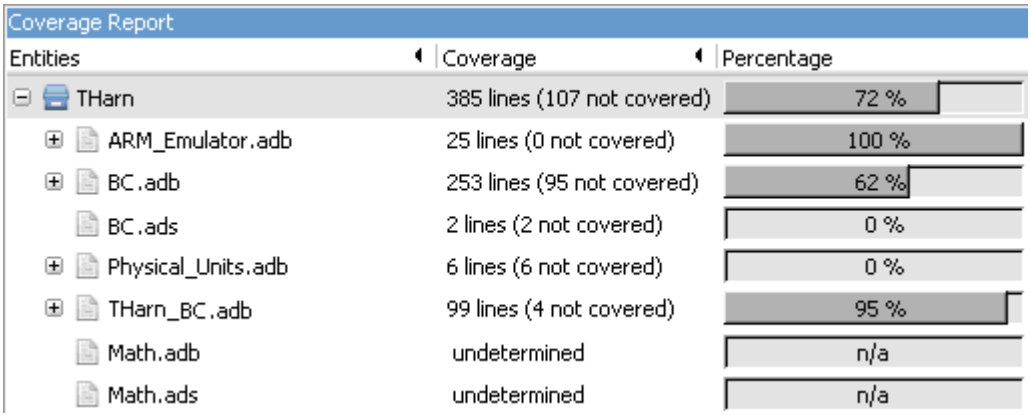


Рис. 11. Пример отчета выполнения теста Coverage. Видно, что модуль ARM_Emulator.adb тестировался полностью, BC.adb – частично, а модуль Math.adb – не участвовал в тестировании.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1. Компиляция программы Ада в среде GNAT

1. Установите на компьютере среду GNAT Programming Studio(например, http://libre.adacore.com/download/configurations > GNAT 2012 > gnat-gpl-2012-i686-pc-mingw32-bin.exe).
2. Создайте рабочий каталог с подкаталогами **src** и **obj**
3. В основной каталог переместите файл проекта **traffic.gpr**

traffic.gpr

```
project Traffic is

   for Object_Dir use "obj";
   for Exec_Dir use ".";
   for Main use ("Traffic.adb");
   for Source_Dirs use (".", "src");

end Traffic;
```

4. В подкаталог **src** скопируйте файл **Traffic.adb**:

Traffic.adb

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Traffic is

  type Airplane_ID is range 1..10;        -- 10 airplanes (= tasks)

  task type Airplane(ID: Airplane_ID);    -- task type representing airplanes
  type Airplane_Access is access Airplane; -- access type (reference) to Airplane

  protected type Runway is             -- a protected object - the shared runway
    entry Assign_Aircraft(ID: Airplane_ID);
    entry Cleared_Runway (ID : Airplane_ID);
    entry Wait_For_Clear;
  private
    Clear: Boolean := True; -- protected private data - generally more than just a flag...
  end Runway;
  type Runway_Access is access all Runway;

  -- the air traffic controller takes requests for takeoff and landing
  task type Controller(My_Runway: Runway_Access) is
    entry Request_Takeoff (ID: in Airplane_ID; Takeoff: out Runway_Access);
    entry Request_Approach(ID: in Airplane_ID; Approach: out Runway_Access);
  end Controller;

  Runway1    : aliased Runway;            -- instantiate a runway
  Controller1: Controller(Runway1'Access); -- and a controller to manage it

 ------ the implementations of the above types ------
  protected body Runway is
    entry Assign_Aircraft (ID : Airplane_ID)
      when Clear is   -- the entry guard - tasks are blocked until this is true
    begin
      Clear := False;     Put_Line (Airplane_ID'Image (ID) &" on runway ");
    end;

    entry Cleared_Runway (ID : Airplane_ID) when not Clear is
    begin
       Clear := True;     Put_Line (Airplane_ID'Image (ID) &" cleared runway ");
    end;

    entry Wait_For_Clear when Clear is begin
       null;
    end;
  end Runway;
```

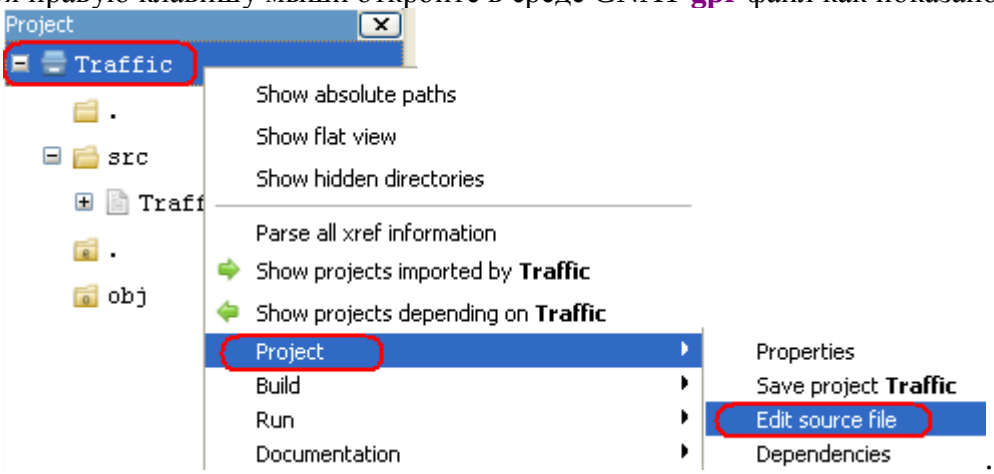
```
  task body Controller is
  begin
    loop
      My_Runway.Wait_For_Clear;   -- wait until runway is available
      select                      -- wait for two types of requests
        when Request_Approach'count = 0 =>  -- landings have priority
         accept Request_Takeoff (ID : in Airplane_ID; Takeoff : out Runway_Access) do
           My_Runway.Assign_Aircraft (ID);  -- reserve runway
           Takeoff := My_Runway;            -- tell airplane which runway
        end Request_Takeoff;                -- end of the synchronised part
      or
        accept Request_Approach (ID : in Airplane_ID; Approach : out Runway_Access) do
          My_Runway.Assign_Aircraft (ID);
          Approach := My_Runway;
        end Request_Approach;
      or                          -- terminate if nobody left who could call
        terminate;
      end select;
    end loop;
  end;

  task body Airplane is
    Rwy : Runway_Access;
  begin
    Controller1.Request_Takeoff (ID, Rwy); -- wait to be cleared for takeoff
    Put_Line (Airplane_ID'Image (ID) &"  taking off...");  delay 2.0;
    Rwy.Cleared_Runway (ID);
    delay 5.0; -- fly around a bit...
    loop
      select   -- try to request a runway
        Controller1.Request_Approach (ID, Rwy); -- this is a blocking call
        exit; -- if call returned we're clear for landing - proceed...

      or delay 3.0;  -- timeout - if no answer in 3 seconds, do something else
        Put_Line (Airplane_ID'Image (ID) &"   in holding pattern");
      end select;
    end loop;
    delay 4.0;  -- do landing approach...
    Put_Line (Airplane_ID'Image (ID) &"            touched down!");
    Rwy.Cleared_Runway(ID);  -- notify runway that we're done here.
  end;

  New_Airplane: Airplane_Access;
begin
  for I in Airplane_ID'Range loop  -- create a few airplane tasks
    New_Airplane := new Airplane(I); delay 3.0;
  end loop;
```

```
end Traffic;
```

5. Загрузите среду **GNAT** и откройте проект **traffic.gpr**. Загрузить среду можно через загрузку файла проекта **gpr**.
6. Используя правую клавишу мыши откройте в среде GNAT **gpr** файл как показано на рисунке: 

. Файл Открытый файл можно редактировать.
7. Откройте программу на языке Ада: Traffic.adb.
8. Просмотрите код программы, найдите отличительные особенности языка Ада и редактора среды GNAT
9. Внесите в код Traffic.adb несколько ошибок разного рода.
10. Настройте стили компилятора как показано на Рис.4.
11. При помощи компилятора найдете и исправьте ошибки в программе.
12. Скомпилируйте исполняемый (exe) файл.
13. Выполните exe файл запуская его в среде GNAT, в Windows и в командной строке Windows. Найдите отличия в выполнении.
14. Настройте GNAT на режим отладки программы (см. соответствующий раздел выше)
15. Запустите отладчик.
16. Установите несколько точек останова в программе (см. Рис. 7).
17. Выведите на экран поля значений нескольких переменных.
18. Выполните программу под отладчиком изменяя его режимы: выполнение до точки останова (breakpoint), выполнение в пошаговом режиме, и др. Постарайтесь определить назначение программы.
19. Выйдете из режима отладки.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначена и как работает программа Traffic.adb.
2. Какие библиотечные функции использует программа.
3. Назовите основные модули интегрированной среды GNAT.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гавва А.Е. – “Адское” программирование. Ada – 95. Компилятор GNAT, V-0.4, май 2004. – 431с., http://www.ada-ru.org/
2. Язык программирования SPARK ADA http://www.adahome.com/Tutorials/Lovelace/lovelace.htm
3. Компилятор GNAT. (Компилятор GCC для Ada интегрирован в GNAT) http://www.gnu.org/s/gnat/
4. Компилятор GCC для Ada http://gcc.gnu.org/