Основные концепции языков программирования

- Обоснованно выбирать язык программирования для реализации конкретного проекта;

- Разрабатывать более эффективные алгоритмы;

- Систематически пополнять набор полезных языковых конструкций;

- Ускорять изучение новых языков программирования;

- Использовать полученные знания как методологическую основу для разработки новых языков программирования;

-Получить базовые знания, необходимые для разработки трансляторов для языков программирования, поддерживающих разные вычислительные модели.

Парадигмы языков программирования

- Императивная; (Algol, BASIC, FORTRAN, PL/1, Ada, Pascal, C, C++, Java)

- Функциональная; (Lisp)

- Декларативная; (Prolog)

- Объектно-ориентированная; (Java, C++, Object Pascal, Smalltalk)

Критерии оценки языков программирования

- Понятность;
- Надежность;
- Гибкость;
- Простота;
- Естественность;
- Мобильность;
- Стоимость.

Понятность

- Уменьшаются требования к документированию проекта, если текст программы является центральным элементом документирования;

- Понятность конструкций языка позволяет легче понимать программу и, следовательно, быстрее находить ошибки;

- Высокая степень понятности конструкций языка позволяет легче сопровождать программу.

Надежность

- Чем раньше при разработке программы обнаружена ошибка, тем меньше стоимость самого проекта;

- Трансляция может быть выполнена на любой машине, воспринимающей входной язык, в то время как тестирование оттранслированной программы должно выполняться на целевой машине либо с использованием программ интерпретации, специально разработанных для тестирования

Гибкость

- Гибкость языка проявляется в том, сколько возможностей он предоставляет программисту для выражения всех операций, которые требуются в программе, не заставляя его прибегать к вставкам ассемблерного кода или различным ухищрениям.

Простота

- Экономия понятий языка предполагает использование минимального числа понятий;
- Ортогональность понятий означает, что между ними не должно быть взаимного влияния;
- Единообразие понятий требует согласованного, единого подхода к описанию и использованию понятий.

Естественность

- Язык должен содержать такие структуры данных, управляющие структуры и операции, а также иметь такой синтаксис, которые позволяли бы отражать в программе логические структуры, лежащие в основе реализуемого алгоритма.

Мобильность

- Язык, независимый от аппаратуры, предоставляет возможность переносить программы с одной платформы на другую с относительной легкостью.

Стоимость

- Стоимость обучения языку;
- Стоимость создания программы;
- Стоимость трансляции программы;
- Стоимость выполнения программы;
- Стоимость сопровождения программы.

Объекты данных в языках программирования

- Имена: идентификатор – строка символов, используемая для обозначения некоторой сущности в программе (переменные, типы, метки, подпрограммы, формальные параметры и др.).

- Константы:

литералы
именованные константы

Объекты данных в языках программирования

- Переменные

Имя,
Адрес,
Значение,
Тип,
Время жизни,
Область видимости

Механизмы типизации

- Статические и динамические типы данных

Недостатки динамического связывания типов:

- Снижается возможность обнаружения транслятором ошибок по сравнению со статическим связыванием типов, т.к. в момент трансляции отсутствует информация о типах переменных;

- При реализации динамического связывания вся информация о типах переменных должна сохраняться в течение всего времени работы программы, что требует значительных дополнительных ресурсов памяти, связанных с необходимостью хранить данные различных типов;

- Динамическое связывание типов приводит к увеличению времени работы программы за счет программной реализации механизмов связывания

Виды типизации

- Слабая

```
Char c;
c=7;

int i;
float x;
…
i=x;

k=x-i;
```

Виды типизации

- Строгая типизация

- Каждый объект данных обладает уникальным типом;
- Каждый тип определяет множество значений и множество операций;
- В каждой операции присваивания тип присваиваемого значения и тип переменной, которой присваивается значение, должны быть эквивалентны;
- Каждая примененная к объекту данных операция должна принадлежать множеству операций, допустимых для объектов данного типа;
- Преобразование типа должно задаваться явно, например i:=integer(x)

Производные типы

```
program sum(input, output);
var
	temp_weight, sum_weight: integer;
	i: integer;
begin
	sum_weight := 0;
	for i:=1 to 10 do
	begin
		read(temp_weight);
		sum_weight:= sum_weight+tem_weight
	end;
	writeln(sum_weight);
end.

// sum_weight:= sum_weight+I;
```

Производные типы

```
program sum(input, output);
type
	weight=integer;
	index=integer;
var
	temp_weight, sum_weight: weight;
	i: index;
begin
	sum_weight := 0;
	for i:=1 to 10 do
	begin
		read(temp_weight);
		sum_weight:= sum_weight+tem_weight
	end;
	writeln(sum_weight);
end.

// sum_weight:= sum_weight+I;
```

Время жизни переменных

- Статические переменные
- Автоматические переменные
- Явные динамические переменные (проблема «висячего» указателя и потерянной динамической переменной)
- Неявные динамические переменные

Область видимости переменных

- Правила видимости переменных определяют, каким образом ссылки на переменные, объявленные вне выполняющейся в данный момент подпрограммы (блока) связаны с объявлениями этих переменных.

Типы данных

- Тип данных – это некоторый класс объектов данных вместе с набором операций для создания и работы с ними. В каждом языке программирования имеется некоторый набор встроенных элементарных типов данных. Дополнительно язык может предоставить возможности, позволяющие программисту определять новые типы данных.

- Под реализацией типа данных понимают
  1) Способ представления объектов данных этого типа в памяти компьютера во время выполнения программы;
  2) Способ представления операций, определенных для этого типа данных (комбинация аппаратных и программных средств, реализующих конкретные алгоритмы и процедуры над представлениями объектов данных заданного типа в памяти).

Числовые типы

- Целый тип (C: int, short, long, char; Pascal: integer, word, longint);
- Вещественный тип (real, float, double)

Логический тип

- bool, boolean – fasle, true

Pascal:

```
var Found: boolean;
begin
      Found:=TRUE;
end.
```

C:

```
bool b=true;
```

Символьный тип и символьные строки

- Char
- String

Перечислимые типы

Pascal:

```
type color = (white, red, green, blue, black);
var circle, square: color;
```

C:

```
enum days {sun, mon, tues, wed, thur,  fri, sat} anyday;
enum gender {man, wom} pol;
anyday=sun;
pol=wom;
if (anyday==0 && pol==wom);
```

Операции, определенные для перечислимых типов:

- сравнение: равно, не равно, больше, меньше, больше или равно, меньше или равно;
- операция присваивания;
- операция succ и pred

Векторы и массивы

Вектор (одномерный массив) – это структура данных, состоящая из фиксированного количества компонентов одного типа. Компонент вектора определяется путем задания индекса, который является целочисленным значением или элементом перечислимого типа

Вектор характеризуется атрибутами:

- количество компонентов (размер вектора);
- тип данных компонентов;
- список значений индексов;

Pascal: var A: array [1..20] of real;

C: char name[20];

Записи

Записью (структурой) называют структуру данных, состоящую из фиксированного количества компонентов (полей), которые могут соответствовать различным типам. Компоненты записей обозначаются символическими именами (идентификаторами).

Запись характеризуется атрибутами:

- количество компонентов записи;
- тип данных каждого компонента;
- идентификатор для каждого компонента

C:

```
struct book
{ char name[20];
  char title[50];
  int year;
float price
} child_book, my_book;
```

Pascal:

```
type Tbook= record
Name: array [1..20] of char;
Title: array [1..20] of char;
Year: integer;
Price: real;
end;
var child_book, my_book: Tbook;
```

Указатели

Указатели включаются в определение языка с целью обеспечения возможности конструирования произвольных структур данных из объектов разного типа.

Над указателями определены следующие операции:

- Операция создания объекта данных фиксированного размера. В памяти отводится место для нового объекта, создается указатель на этот объект, которому присваивается значение ссылки (адреса) на этот объект;
- Операция разыменования – операция использует значение указателя для доступа к объекту данных, на который он ссылается;

```
int a,b;
int *pa;
a=1;
pa=&a;
b=*pa;
```

```
var pa: ^integer;
a,b: integer;
begin
a:=1; pa:=@a;
b:=pa^;
end.
```

Выражения и операторы присваивания

Арифметические выражения состоят из операндов, операторов, круглых скобок и вызовов функций.

Порядок вычислений определяется приоритетом операторов и может нарушаться при помощи использования скобок.

Возможность появления побочного эффекта от использования функций типа X=FUNC(X);

C:

```
COUNT, SUMMA=0;
COUNT+=1;
SUMMA=++COUNT;
SUMMA=COUNT++;
```

Pascal:

```
COUNT:=0; SUMMA:=0;
COUNT:=COUNT+1;
```

Выражения и операторы присваивания

Логические выражения состоят из логических операндов (переменных, констант, вызовов функций, возвращающих результат логического типа, и выражений отношения), круглых скобок и логических операторов.

Наиболее часто используются логические операторы И, ИЛИ, НЕ.

Проблемы сокращенного вычисления, связанные с побочным эффектом.

C:
(A>=0) && (B<0)

Pascal:
(A>=0) and (B<0)

Структуры управления на уровне операторов

- Композиция – операторы могут быть представлены в виде последовательности, выполняемой как единое целое;
- Ветвление – две последовательности операторов могут быть альтернативными, при этом в каждом конкретном случае выполняется один из них;
- Повторение – последовательность операторов может выполняться многократно или вообще не выполняться в зависимости от некоторого условия.

Составной оператор (блок)

Pascal:

```
begin
      оператор 1;
      оператор 2;
…
      оператор n
end;
```

C:

```
{
      оператор 1;
      оператор 2;
…
      оператор n;
}
```

Операторы if

Pascal:

if логическое выражение then оператор;

if логическое выражение then оператор else оператор;

C:

if (логическое выражение) блок else блок;

if (логическое выражение) блок else if (логическое выражение) блок else блок;

Переключатели

Pascal:

```
case переменная of
список констант 1:
оператор;
список 2: оператор;
else
оператор
end; { case }
```

C:

```
char n;
switch (n)
{case ‘a’:
оператор 1;
оператор 2;
break;
case ‘b’:
оператор 1;
оператор 2;
break;
default:
оператор 1;
оператор 2;
break;
}
```

Цикл while (while - do)

Pascal:

while логическое
выражение do
тело цикла;

C:

while (условие цикла)
do {тело оператора}

Цикл repeat (do - while)

Pascal:

Repeat
тело цикла
until логическое
выражение;

C:

do {тело оператора}
while (условие цикла);

Цикл for - do

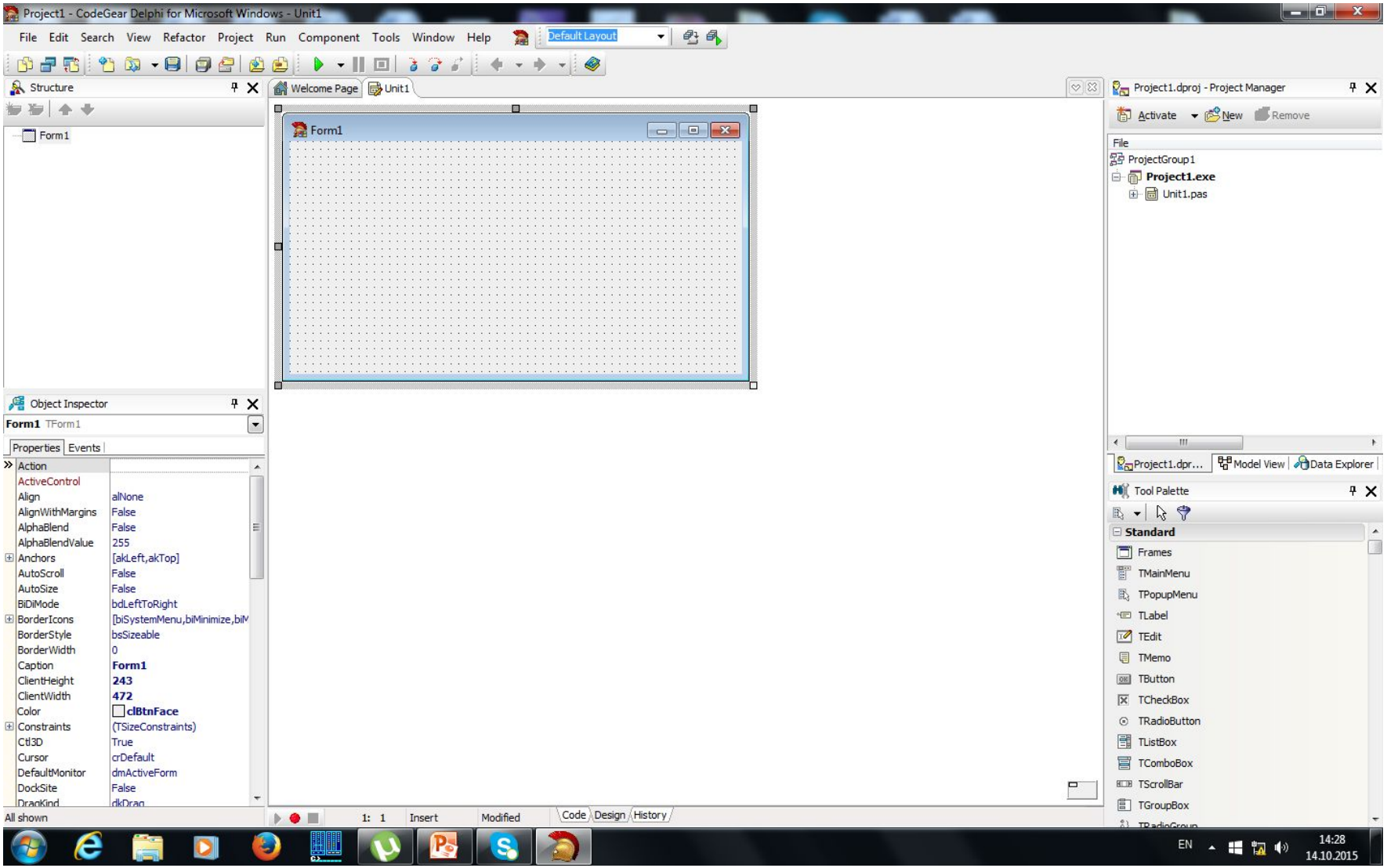
Pascal:

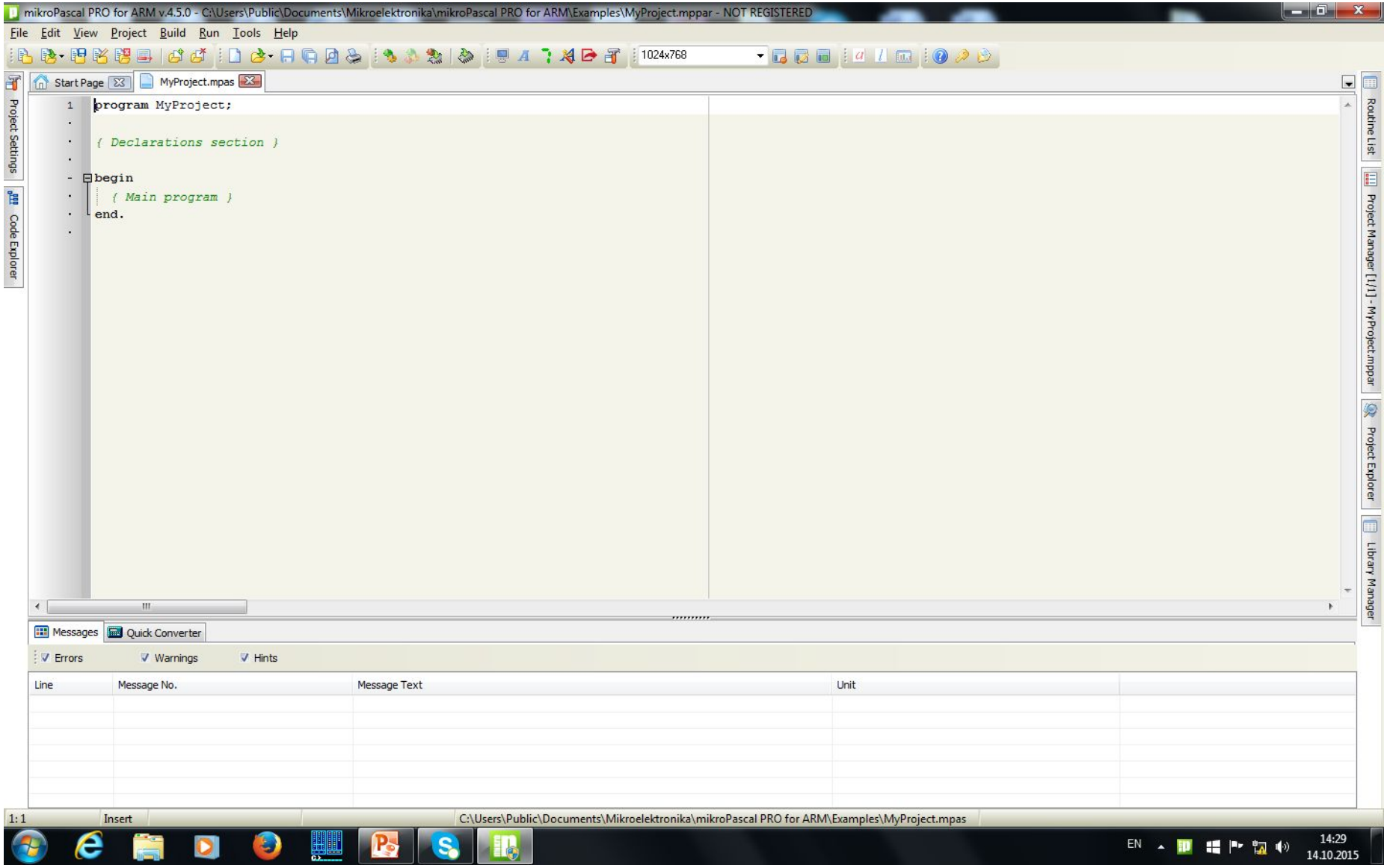
```
for индексная
переменная:=
начальное значение
to конечное значение
do
тело цикла;
```

C:

```
for (стартовое значение
переменной цикла; условие
продолжения; изменение шага
переменной цикла)
тело цикла;
```

```
int c, i;
char s[10];
c=64;
for (i=0; i<10; i++) do s[i]=c;
```


Project1 - CodeGear Delphi for Microsoft Windows - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
Default Layout
Structure
Form1
Welcome Page
Unit1
Form1
Project1.dproj - Project Manager
Activate
New
Remove
File
ProjectGroup1
Project1.exe
Unit1.pas
Project1.dpr...
Model View
Data Explorer
Object Inspector
Form1 TForm1
Properties
Events
Action
ActiveControl
Align alNone
AlignWithMargins False
AlphaBlend False
AlphaBlendValue 255
Anchors [akLeft,akTop]
AutoScroll False
AutoSize False
BiDiMode bdLeftToRight
BorderIcons [biSystemMenu,biMinimize,biM
BorderStyle bsSizeable
BorderWidth 0
Caption Form1
ClientHeight 243
ClientWidth 472
Color clBtnFace
Constraints (TSizeConstraints)
Ctl3D True
Cursor crDefault
DefaultMonitor dmActiveForm
DockSite False
DragKind dkDrag
All shown
Tool Palette
Standard
Frames
TMainMenu
TPopupMenu
TLabel
TEdit
TMemo
TButton
TCheckBox
TRadioButton
TListBox
TComboBox
TScrollBar
TGroupBox
1: 1
Insert
Modified
Code
Design
History
EN
14:28
14.10.2015


mikroPascal PRO for ARM v.4.5.0 - C:\Users\Public\Documents\Mikroelektronika\mikroPascal PRO for ARM\Examples\MyProject.mppar - NOT REGISTERED
File Edit View Project Build Run Tools Help
1024x768
Start Page
MyProject.mpas
1 program MyProject;
{ Declarations section }
begin
{ Main program }
end.
Project Settings
Code Explorer
Routine List
Project Manager [1/1] - MyProject.mppar
Project Explorer
Library Manager
Messages
Quick Converter
Errors
Warnings
Hints
Line
Message No.
Message Text
Unit
1: 1
Insert
C:\Users\Public\Documents\Mikroelektronika\mikroPascal PRO for ARM\Examples\MyProject.mpas
EN
14:29
14.10.2015