اذا الى الان عرفنا كيف تم بدا كتابة البرامج وبينت من خلال الشروحات السابقةالحاجة الى استعمال البرمجة التركيبية او الاجرائية وقد عرفنا ان البرمجة المهيكلة ليست سوى اسلوب وجب على المبرمجين اتباعه وبينت ان هذا الاسلوب يهدف الى كتابة برامج بطريقة معينة ومرتبة والامتناع عن استخدام تعليمات ال goto وذالك لتحسين مقرؤية البرنامج وجعله اسهل لفهم

وايضا تحدثنا عن تطور النظم البرمجية الى ان وصلنا الى الاغراض واعطينا وصفا عاما وشرحا جد مختصر عنها وبينا مالحاجة لاستخدامه وذالك لتسهيل فهم البرمجة وعملها لدى المبرمج

لنتابع

المتغيرات

وصلتني عدة رسائل من اخوة طالبين فيها تعلم الدلفي ..... وقد ذكر في بعضها انهم يريدون معرفة المتغيرات والية عملها ..... رغم انهم مهندسون في مجال الحاسوب ....؟؟؟الى هؤلاء وكل من سوف يقرأ هذه الدروس اقول تفضل هذا شرح جد مختصر عن المتغيرات ارجو ان يقدم بعض من الفائدة

مثال في مثالنا السابق يمكن كتابة البرنامج على هذا الشكر

```
var
  s:string;
begin
  { TODO -oUser -cConsole Main : Insert code here }
  s:=('Hello World');
  writeln(s);
end.
```

ويكون لدينا نفس خرج البرنامج.....؟؟؟؟

ماهو المتغير ولماذا نحتاجه وكيف يعمل كيف يكتب لماذا وكيف ومالى هنالك من اسئلة

المتغير هو رمز الى موضع في ذاكرة الحاسب وكل متغير له اسم **name**

ونمط او نوع**type**

وحجم **size**

وقيمة **value**

فالمتغير variable

هو مكان او موضع في الذاكرة يقوم المجمع بحجزه

ففي مثالنا السابق

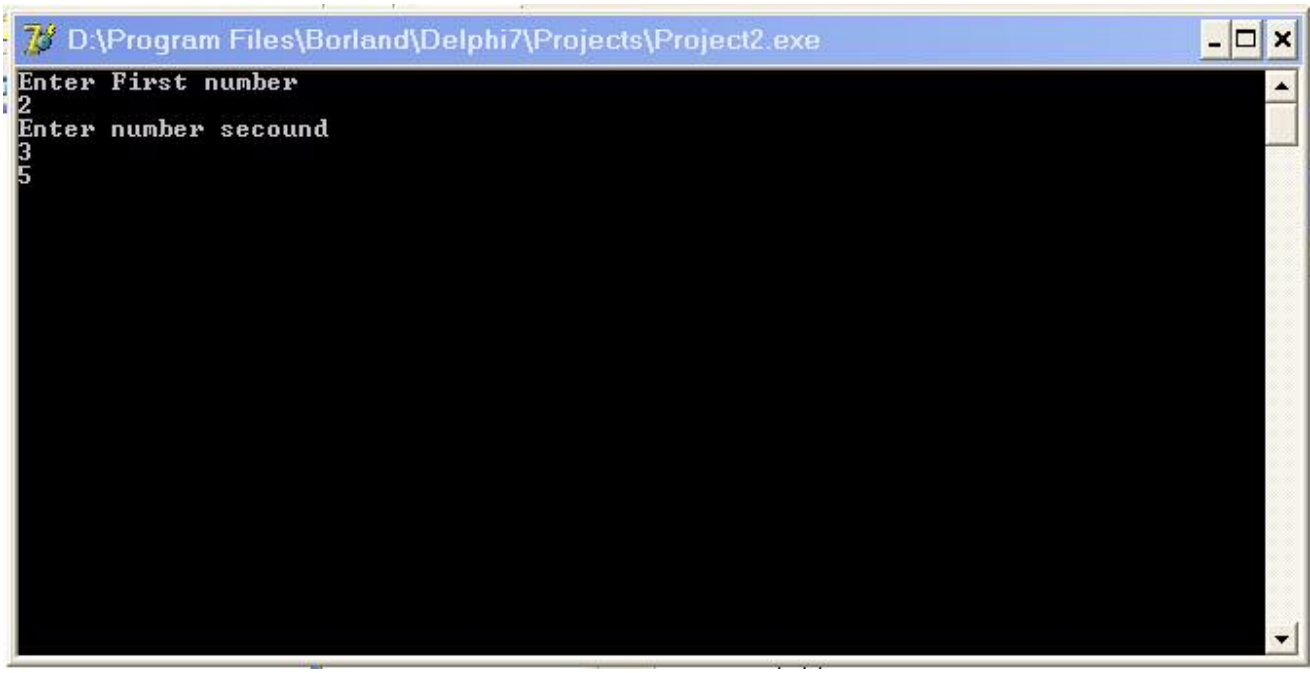
اخبرنا ان المتغير **s**هو من نوع سلسلة حرفية.....

لنترك هذا البرنامج الازلي **HelloWorld**بشأنه ولنتابع

مثال اخر جمع رقمين

Project1

```
uses
  SysUtils;
var
  x,y,z:integer;
begin
 writeln('Enter First number');
 Readln(x);
 writeln('Enter number secound');
 Readln(y);
 z:=(x+y);
 write(z);
end.
```



قلنا ان المتغير هو موقع او مواضع من الذاكرة فالذاكرة عبارة عن غرف او خانات عديدة يقوم الحاسب بترقيمها او تسميتها كل خانة على حدة وعندما صرحنا عن المتحول في المثال السابق وهو مثلا **x** وهو من نوع رقم اخبرنا المترجم اننا نريد حجز مكان بالذاكرة لنضع فيه رقما من-32768 الى +32768 طبعا وهذه القيم معروفة مسبقا من قبل المجمع وماعليك انت كمبرمج الا ان تصرح عنها .....

تنويه5 في مثالنا السابق قمنا بالتصريح عن عدة متغيرات او متحولات من نوع **integer** اذ لايشترط ان يتم حجز مواضعها بالذاكرة متجاورين

تنويه6 قيمة هذا المتغير تم تحديدها من قبل المرمج خلال تصريحه فالبرنامج لن يقبل باحرف يسبب ادخال الاحرف والضغط على الانتر ينتج رسالة خطأ

تنويه7 يجب التصريح عن المتغيرات قبل استخدامها

مثال في مثالنا السابق المتغير **Z** لايمكن استخدامه ان لم نقم بالتصريح عنه بانه متغير من نوع رقمي اي **integer**

شرح البرنامج

الامر **read ,readln**

اذا كان الامر **write or wirteln** يقومان بطباعة على الشاشة فان الامرين **read or readln**يقومان بقرائة ماتم كتابته

طبعا والفارق بين **read and readln** هو نفسه الفارق بين **write and writeln**

في البداية قمت بالتصريح عن متحولات **z,s,x**من نوع رقمي **integer**

بعد ذالك السطر

writeln('Enter First number');

وهو لطباعة السلسة المحرفية

تنويه8 ان الفراغ او الازاحة العمودية يعتبرها المجمع هنا حرف مثلها مثل اي حرف من السلسة يقوم المجمع بطباعتها كما هي كما هو موضح بالمثال السابق

السطر التالي والذي هو

readln(x);

اي اقرأ القيمة المدخلة واعتبرها او اسندها او هي التي سوف تكون قيمة **x**

السطران التاليان نفس الشرح

اما السطر الخامس

z:=(x+s);

هو اعطاء قيمة **z**او اسناد قيمته وهي **x+s**

السطر الاخير وهو طباعة النتيجة

تنويه9 من الملاحظ ان المجمع يتجاهل الحروف الواقعة ضمن الفاصلتان العلويتان

مثال :ففي المثال وفي السطر الاخير

z := (x+s);

وقمنا بوضع الفاصلتان العلويتان على الشكل التالي

z := ('x+s');

فان خرج البرنام سوف ينتج خطأء قواعديا وذالك لانه اعتبرها سلسلة حرفية وقد كنا صرحنا عنه انه رقم **integer**

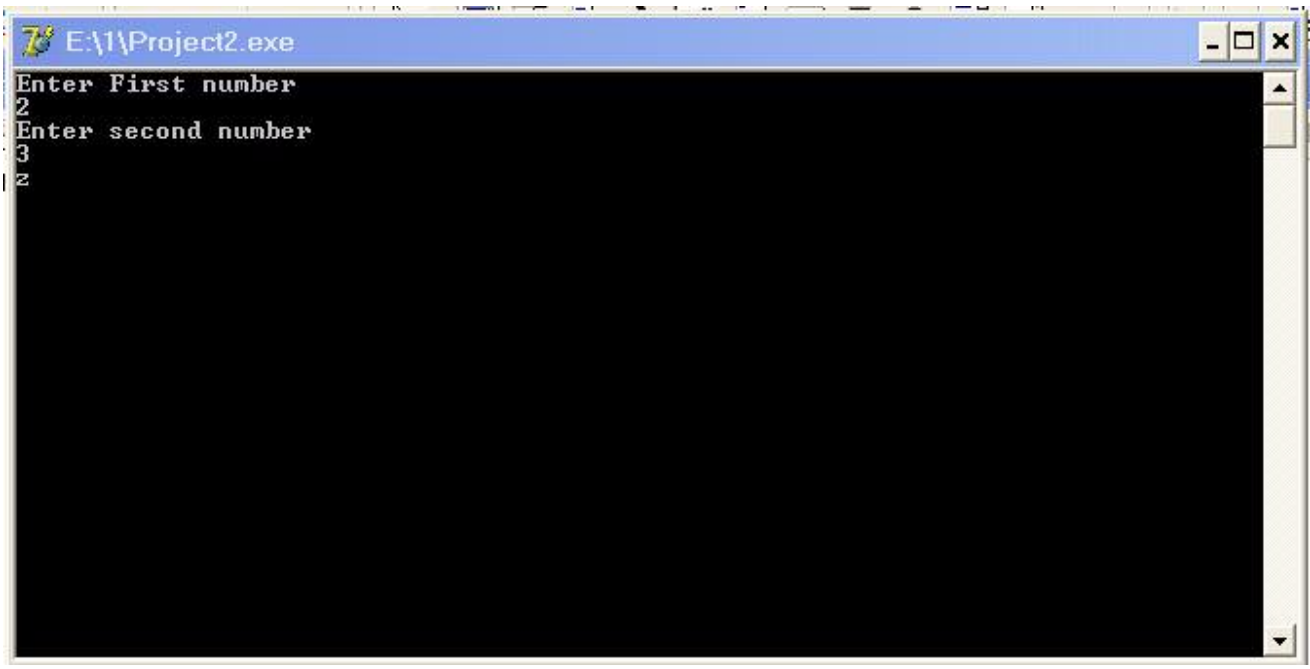
او لو قمنا بتغير التعليمة الاخيرة

writeln (z);

ووضعنا الفاصلتين العلويتين على الشكل

writeln ('z');

فان النتيجة الخرج سوف تكون فقط **z** انظر



لانه وكما قلت تم طباعة **z**دون النظر الى قيمتها

تنويه10 من المفضل وضع فراغا بعد كل اشارة **,** لتحسين مقرؤية البرنامج

مثال:

في مثال البرنامج كان الافضل الكتابة بدلا من

z,s,x:integer;

كان علي كتابته على الشكل

z , s , x : integer ;

ايهما اسهل بالقرائة ويدعم الاسلوب المهيكل ...؟؟؟؟

طبعا هذه الفكرة نوه واكد عليها ماركو من خلال الشرح الوارد في الدرس السابق

تنويه11 يفضل بعض المبرمجين التصريح عن كل متحول ضمن سطر منفصل ..... تساعد هذه الصيغة على اضافة تعليق الى جانب كل تصريح وبالتالي تحسين من مقرؤية البرنامج

مثال : في مثالنا السابق وبدلا من التصريح كما ورد

z , s , x : integer ;

يمكن التصريح على الشكل

z : integer ;

s : integer ;

x :integer ;// هنا يكتب التعليق مثلا

تنويه12 قيمة هذا المتحول هذه فقط لمنظومة ال 16 بت اما بنسخ الدلفي ابتداء من الرابعة فغير محددة

كما راينا مرت معنا التعليميتين = و+ وهما تعليمتا اسناد كما وردو وتسمى العمليتين (+)و(=) بعمليتيتين ثنائيتين **Binry operators** لاني اي عملية منهما تحتاج الى معاملين **two operators** ففي الحالة الاولى يشكل التحولان **x,s**المعاملين الخاصين بها اما في حالة = فيشكل المتحول **z** وقيمة **(x+s)** المعاملين الخاصين بتلك العملية

تنويه13 تتم معظم الحسابات باستخدام تعليمات الاسناد

كيف جرى تنفيذ البرنامج

ففي البرنامج السابق عندما جرى تنفيذ التعليمة

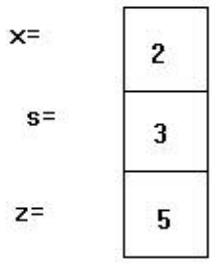
readln(x)

وقام المستخدم بادخال القيمة فان الباسكال تضعها في موضع الذاكرة المخصص(الخانة التي حجزت لهذا المتحول) ففي مثالنا قمنا بادخال القيمة (2) كقيمة للمتحول **(x)** فسيقوم الباسكال بوضع القيمة (2) في الموضع المحجوز للمتحول **(X)**

ونفس الامر سوف يحصل بالنسبة للمتحول **(s)**

وبعد ان يحصل البرنامج على قيمة المتحولين التي قام المستخدم بادخالهما فانه يقوم بعملية الجمع ويضع النتيجة في المتحول **(Z)**

وبعد القيام بعملية الجمع تبدو الذاكرة مثلا على الشكل التالي



تنويه14 لاحظ بقاء موضعي الذاكرة للمتحولين **S,X**على شكلها السابق بعد القيام بعملية الجمع المرتبطة بالمتحول او المتغير **z** حيث تم استخدام قيهما ولكن لم يجري تدميرهما اذا يمكننا ان نقول انه لاتؤدي عملية قرائة قيمة مخزنة في موضع الذاكرة الى تدمير تلك القيمة او ضياعها

تنويه15 اذا كان قيمة المتحول **x**مثلا معلوماً او يوجد قيمة في موضعه في الذاكرة قبل تنفيذ هذا البرنامج ماذا يحصل لن يحدث شيئ فقط ان القيمة المدخلة الان في برنامجنا السابق سوف تقوم بتدمير اي قيمة كان تشغل موضع المتحول **x**في الذاكرة مثال

لنضيف على البرنامج التالي تعليمة وهي اعطاء قيمة للمتحول **x**كما يلي

```
uses
  SysUtils;
  var
  s,z,x:integer;
begin
  { TODO -oUser -cConsole Main : Insert code here }
  writeln('Enter First number');
  x:=(10);
  readln(x);
  writeln('Enter second number');
  readln(s);
  z:=x+s;
  writeln('z is',z);
end.
```

هنا في هذا البرنامج قمنا باعطاء قيمة للمتحول **x** نفذ البرنامج ماذا ترى

همهم كما ترى ان خرج البرنامج لم يتغير ابدا لماذا .....؟؟؟؟

كما قلت في الدرس الثالث ..... اعد قرائته جيدا وهذه المرة بتمعن اكثر

ان لغات البرمجة تنفذ التعليمات او الاوامر بالتسلسل واحدة تلوة الاخرى حسب ورودها في نص البرنامج طبعا ان لم ننقل بنية التحكم باستخدام تعليمة ال **goto** وهنا اتت التعليمة والتي قمنا باعطاء قيمة للمتحول **x** قبل تعليمة **readln(x)**

ولذالك فان البرنامج وقت التنفيذ عندما وصل للتعليمة **x:=(10);**

فانه ادخل قيمة **x** في موقعها بالذاكرة ولكنه عندما وصل للتعليمة **readln(x);**فانه انتظر المستخدم لاعطاء قيمة للمتحول **x**وعندما قام المستخدم باعطاء تلك القيمة فان الباسكال او الدلفي قام بتدمير القيمة الاولى المسندة ووضع محلها القيمة الجديدة وهنا نستنتج ان قيمة المتغير المصرح به تتغير باستمرار كما راينا بالمثال السابق التوضيحي

مثال اخر لنضيف تلك التعليمة الجديدة وهي **x:=(10);**

بعد التعليمة **readln(x)** انظر ماالنتيجة .....

```
readln(x);

readln(s);

x:=(10);

z:=(x+s);

writeln(z);
```

انظر الخرج

ارجو ان تكون قد تكوَن لك مفهوما عاما وواضحا عن المتغيرات

الان لاحظنا انني اخترت اسمائا للمتحولات في الامثلة السابقة هي **z,x,s**

ولكننا نستطيع ان نسمي متحولاتنا بالاسم الذي نشاء هذا ماتبيحه لنا الدلفي طبعا وبالطول الذي نشاء

ولكن هناك كما لغات البرمجة الاخرى يجب مراعاتها في اسم المتحول

اولا يجب الانتباه على الا يبدا اسم المتغير برقم مثال

2zaher:integer;

خطأ لانه بدأ برقم ممكن ان نسميه فقط

zaher: integer;

يجب ان يحوي اسم المتغير على ارقام وحروف فقط مثال

zaher1 :integer;

ملاحظة لقد وضعت الاقواس هنا من اجل تحسين مقرؤية البرنامج فقط اي يمكن ان تكتب البرنامج بدون اقواس طبعا فقط لقيم الاسناد

zaher2 :integer;

zaher3:integer;

هذه كانت امثلة يقبل بها المجمع اما

zah er1 :integer;

فلايقبل بها لوجود فراغ بالاسم على سبيل المثال طبعا

ثالثا يجب الا يكون الاسم من ضمن الكلمات المحجوزة والمعرفة سابقا لدى المجمع مثال

integer : integer ;

خطأ لانه كلمة محجوزة او مثلا

string :char ;

ايضا

اما الكلمات المحجوزة فاليك بعضا منها نقلا عن ماركو حرفيا حيث يقول

الكلمات المفتاحية

الكلمات المفتاحية هي كل المعرّفات المحجوزة من قبل اوبجكت باسكال، و التي لها دور في اللغة. دليل دلفي (Help) يميّز بين الكلمات المحجوزة والتوجيهات كالتالي: الكلمات المحجوزة لا يمكن استخدامها كمعرّفات، بينما التوجيهات لا يجب استخدامها لنفس الغرض، حتى لو قبلها المجمّع. عند الممارسة، عليك تجنّب استخدام أية كلمة محجوزة كمعرّف. (او كتصريح ....تنويه)

في الجدول 2.1 يمكنك رؤية قائمة كاملة بالمعرّفات التي لها دورا خاصا في لغة اوبجكت باسكال (في دلفي 4)، بما في ذلك الكلمات و الكلمات المحجوزة الأخرى.

الجدول 2.1: الكلمات المفتاحية و الكلمات المحجوزة الأخرى في لغة اوبجكت باسكال اي الدلفي

الكلمة الرئيسة	الدور
absolute	directive (variables) موجّه (متغير)
abstract	directive (method) موجّه (مسار)
and	operator (boolean) معامل (بولي)
array	type نوع
as	operator (RTTI) معامل
asm	statement تعليمة
assembler	backward compatibility (asm) توافقية مع السابق
at	statement (exceptions) تعليمة (استثناءات)
automated	access specifier (class) محدد دخول (طبقة)
begin	block marker علامة حيّز
case	statement تعليمة
cdecl	function calling convention قاعدة استدعاء وظيفة
class	type نوع
const	declaration or directive (parameters) تعريف أو توجيه (محددات)
constructor	special method مسار خاص
contains	operator (set) عامل (فئة)
default	directive (property) توجيه (سمة)
destructor	special method مسار خاص
dispid	dispinterface specifier واجهة اطلاق محدد
dispinterface	type نوع
div	operator عامل
do	statement تعليمة
downto	statement (for) تعليمة (for)
dynamic	directive (method) توجيه (مسار)
else	statement (if or case) تعليمة (if او case)
end	block marker تعليم حيّز
except	statement (exceptions) تعليمة (استثناءات)
export	backward compatibility (class) توافقية مع السابق
exports	declaration تعريف
external	directive (functions) توجيه
far	backward compatibility (class) توافقية مع السابق
file	type نوع
finalization	unit structure بنية وحدة
finally	statement (exceptions) تعليمة
for	statement تعليمة
forward	function directive وظيفة توجيه
function	declaration تعريف

statement تعليمة	goto
statement تعليمة	if
unit structure بنية وحدة	implementation
directive (property) توجيه (مسار)	implements
operator (set) - project strucure عامل (فئة) - بنية مشروع	in
directive (dipinterface) توجيه	index
statement تعليمة	inherited
unit structure بنية وحدة	initialization
backward compatibility (see asm) توافقية مع السابقة	inline
type نوع	interface
operator (RTTI) عامل	is
declaration تعريف	label
program structure برنامج بنية	library
directive (method) توجيه (مسار)	message
operator (math) عامل (رياضي)	mod
directive (function) توجيه (وظيفة)	name
backward compatibility (class) توافقية مع السابق	near
value قيمة	nil
directive (property) توجيه (مسار)	nodefault
operator (boolean) عامل (بولي)	not
backward compatibility (class) توافقية مع السابق	object
statement (case) تعليمة	of
statement (exceptions) تعليمة	on
operator (boolean) معامل (بولي)	or
directive (parameters) توجيه (محدّدات)	out
function directive وظيفة توجيه	overload
function directive وظيفة توجيه	override
program structure (package) بنية برنامج (حزمة)	package
directive (record)(تسجيلة) توجيه	packed
function calling convention استدعاء وظيفة طرقة	pascal
access specifier (class) معيّن لوصول (طبقة)	private
declaration تعريف	procedure
program structure برنامج بنية	program
declaration تعريف	property
access specifier (class) معيّن لوصول (طبقة)	protected
access specifier (class) معيّن لوصول (طبقة)	public
access specifier (class) معيّن لوصول (طبقة)	published
statement (exceptions) تعليمة (اعتراضات)	raise

property specifier سمة معيّن	read
dispatch interface specifier معيّن واجهة ارسال	readonly
type نوع	record
function calling convention طريقة لاستدعاء وظيفة	register
function directive وظيفة توجيه	reintroduce
statement تعليمة	repeat
program structure (package) بنية برنامج (حزمة)	requires
directive (functions) توجيه (وظيفة)	resident
type نوع	resourcestring
function calling convention طريقة لاستدعاء وظيفة	safecall
type نوع	set
operator (math) عامل (رياضة)	shl
operator (math) عامل (رياضة)	shr
function calling convention طريقة لاستدعاء وظيفة	stdcall
directive (property) توجيه (سمة)	stored
type نوع	string
statement (if)	then
declaration تعريف	threadvar
statement (for) تعليمة	to
statement (exceptions) تعليمة (استثناءات)	try
declaration تعريف	type
unit structure بنية وحدة	unit
statement تعليمة	until
unit structure بنية وحدة	uses
declaration تعريف	var
directive (method) توجيه (مسار)	virtual
statement تعليمة	while
statement تعليمة	with
property specifier سمة معيّن	write
dispatch interface specifier معيّن واجهة ارسال	writeonly
operator (boolean) عامل (بولي)	xor

تنويه16

وكما تعليمنا من اسلوب البرمجة المهيكلة لذالك ينصح ان يكون المصرح به يدل على المضمون لتسهيل مقرؤية البرنامج بدلا من تسمية اسماء غير مفهومة تتطلب العودة للمرجع البرنامج او استخدام التعليقات بشكل زائد عن الحد

طول الاسم :ن الباسكال بشكل عام او الدلفي لاتقيد طول معين لعدد حروف الاسم يعني مثلا يمكن ان تكتب اسم متغير يبلغ طوله مائة حرف ورقم مثلا غالبا حتى 255 مسموح بها بالدلفي هذه المعلومة لم اجربها الى الان لاتاكد من صحتها ولم اسمع انه تسبب في اي اشكال

ولكن ينصح بالا تستخدم اكثر من 20 وكحد اعظمي 31 حرف كما في السي++ لتحسين المحمولية

نسمي ال **integer**بنمط او نوع البيانات المراد ادخالها وايكم بعضا من انواع البيانات ا كما اذكرها وهي قيم تقريبية غالبا

نمط المعطيات	الحجم بالبايت	مجال المسموح به
shortInt	1	128- to +128
Byte or char	1	0 to 255
Woed or Widechar	2	0 to 65535
longInt or Integer	4	-2147483648 to+ same number
Int64	8	-10,000,000,000,000,000,000 to + same number
Cardinal	4	0 to 2147500000
Single	4	1,5 x 10(-45) to 3,4 x 10(+38) البين اقواس هي اس أي قوة العدد
Double or Real	8	5 x 10(-324) to 107x 10 (309)
Extended	10	3,5 x 10(-5000) to 1,1 x 10(5000)
Comp	8	تقريبا نفس الا int64
currecy	-8	تقريبا نفس السابقة
Boolen	1	true or False
Variant	16	Varies
smallint	2	-32768 to +32768

تنويه17 نلاحظ من الجدول ان قيمة **integer and LongInt** متطابقين اذا لماذا يوجد في الدلفي نمطي او نوعي بيانات او معطيات مختلفين وهما متشابهين تماما ....؟؟؟

كان ذالك في النسخ القديمة للدلفي حيث في بيئات البرمجة 16 بت يتطلب **Integer 2**بايت للتخزين و **LongInt** 4 بايت للتخزين اما في بيئة ال **win32** كلاهما يحتاج الى 4 بايت للتخزين وبالتالي لديهما نفس القيمة ابتداء من الدلفي 4 وكما ارى ان معظم المبرمجين يستخدمون **integer**بدلا من الثاني

كما يمكن ان نلاحظ ايضا انه لانماط **Int64 and Camp** ايضا قيم متطابقة والفرق يكمن في طريقة معالجة المجمع لهما داخليا فالاول هو من نمط **Integer**اما الثاني فهو من نمط **REAl**

انا من جهتي لم يلزمني ان استخدم النمط الثاني الى الان ولكن يمكن ان يكون هناك سبب منطقي لتستخدمه انت ببرامجك

كما ايضا نرى ان النمط **Real and double** متطابقين ذالك انه كان في النسخ السابقة بالدلفي

فقد كان ياخذ النمط **real**6 بايت اما الان هو ياخذ 8 بايت ليصبح متوافقا مع العالجات الحالية ويعتبر قديما واغلب المبرمجين يستخدمون النمط **Double**عوضا عنه

طبعا لم اقم بادخال انواع او انماط السلاسل الان سوف ياتي على شرحها لاحقا

تنويه18 من الجدير ذكره هو ان الانماط **Currency,Extended,Double,Single** تستسخدم ارقام الفاصلة العائمة (يعني رقم مع جزء عشري مثلا3.77) بقية الانماط البيانات تتعامل فقط مع القيم الصحيحة اي بدون فواصل

لاتهتم لذالك لان المجمع سوف يخبرك ماذا يمكنك فعله وماذا لايمكنك فعله في هذا الصدد

مسالة دراسية:1

مثلا البرنامج التالي :

```
var
  Integer1 : SmallInt ;
  Integer2 : SmallInt ;
  Integer3 : SmallInt ;
begin
  { TODO -oUser -cConsole Main : Insert code here }
  Integer1 := (200) ;
  Integer2 := (200) ;
  Integer3 := (Integer1 * Integer2) ;
  Writeln (Integer3) ;
end.
```

خرج البرنامج هو كالاتي

```
-25536
```

لماذاااا ..... ؟؟؟؟

ان لم تعرف اعد قرائة الدرس بتمعن وتركيز اكبر

ان القيمة الكبرى لنوع المعطيات **smallInt**هو 32768 وهنا حاصل الجداء هو 40000 فلماذا لم يظهر لنحاول فهم الامر

تنويه19 لاحظ الارشاد الذي يظهر لك عندما عرفت المتحول كيف ان الدلفي اخبرك بالقيمة العظمى والدنيا لهذا النوع من المتغير هذا ميزة جيدة للدلفي ان لم تنتبه عليها فلاتحزن ماعليك سوى اعادة مؤشر الفأرة على التابع ليظهر لك الارشاد من جديد

تنويه20 اعيد واكد ان الاقواس هنا فقط لتحسين المقرؤية اي يمكنك كتابة الشيفرة بلا اية اقواس سوف ناتي على ذكر هذه الاقواس لاحقا لاتتعجل

ماذا يحصل ان اضفنا لل 32768 واحد من نفس نوع المتحول سوف تحصل على قيمة -32768 هذا تماما مايحدث في عداد السيارة عندما يكون على قيمة 999و99 وعندما يقود السائق فان العداد سوف سقلب الى النتيجة 000و00 تعالو لنجرب فعندي وعندكم الدلفي ومازلنا نتعلم همهم

انظر

```
var
  Integer1 : SmallInt ;
  Integer2 : SmallInt ;
  Integer3 : SmallInt ;
begin
  Integer1 := (32767) ;
  Integer2 := (2) ;
  Integer3 := (Integer1 * Integer2) ;
  Writeln (Integer3) ;
end.
```

نفذ الان البرنامج وانظر ما النتيجة

هذا المثال يشرح ماهو معروف باللف لذالك يجب ان تكون يقظا وتتنبه لاختيار متغيراتك وان تختار النوع الكبير بما يكفي لتحوي المتغيرة القيمة دون اي لف او فائض عن القيمة المحددة لها

من جهتي انا غالبا ماستخدم النمط **Integer**

هنا ذكرنا مثالا استطاعت الدلفي ان تتغاضى عنه وقام المجمع لديها بتصريفه مع ثلاث رسائل انذار دون ان توقف التصريح في اول تصريف له وذهبت الرسائل مع التصريفات الاتية

تعالو لنختبر

```
var
  x1 : Byte ;
begin
 x1 := (300);
end.
```

هنا عجزت الدلفي او الباسكال عن التصريف عندما اردنا تصريف المشروع واصدرت رسالة خطأ من المجمع يقول

Error] Unit1.pas(29): Constant expression violates subrange bounds]

ترجمتها الحرفية هي

تعبير ثوابت تتعدى حدود المجال المسموح

حيث كما نرى يخبرنا المجمع اننا لانستطيع إسناد قيمة 300 الى المتحولة **x** والسبب لانه تم التصريح عن معطيات او نوع هذه المتغيرة من نوع او نمط **Byte** وهذه المتغيرة تاخذ فقط من 0 الى 255

تنويه21 ان اردت ان تصبح مبرمجا متمرسا بالدلفي فاخرص على معالجة ارشادات وتنبيهات او رسائل المجمع كاخطاء او حتى لو تم التصريف .... لماذا

لان المجمع يحاول اخبارك ان هناك شيئا ما ليس صحيحا تماما في شفرتك ويجب عليك كمبرمج ان تحترم هذه التصريحات والرسائل وعليك ان تجتهد بتصريف الشيفرة دون تحذيراات ولكن في حالات

نادرة سوف ترى انه لايمكنك تجنب تحذير المجمع ولكن مع ذالك كن متيقنا من انك رايت كل التحذيرات وانك تعرف ماهيتها مع ذالك وحاول تصحيحه ماامكن

تنويه22 لاحظو اني استخدمت في هذا المثال تصريح **integer1**وهذا جائز لاني هذه المحرفية ليست مجوزة

تنويه23 لاحظو اني قد تلافيت الملاحظات التي ظهرت علي بكتابة البرامج والتي كنت لااحترم او اكتب بالاسلوب المهيكل ..... لذالك ادعوكم على ذالك اليس البرنامج اصبح و اسهل مقرؤية ...؟؟؟

من الاستخدامات الاخرى ايضا لكلمة المفتاحية **Var** تستخدم للتصريح عن بارامترات التوابع والاجرائيات كمتغيرات بارامترية سوف نتعلم سويا كيف في المحاضرات تنبيه محاضرات لنتذكر اننا مازلنا بالمحاضرة الاولى اذا بالمحاضرات القادمة باذن الله

همهم لنورد مثال اخير لنتاكد اننا فهمنا ماهو المتغير او المتحول والية عمله وكيف يجب ان نتعامل معه وان نستثمره في مشاريعنا وبرامجنا

مسألة دراسية2 انظر البرنامج التالي

```
var
  num1 : Integer ;
  num2 : Integer ;
begin
 num1 := (10) ;
 num1 := (num1 + 11) ;
 num2 := (5) ;
 num2 := (num2 + 5) ;
 num1 := (num1 + num2) ;
 writeln (num1) ;

end.
```

تعالو لنستنتج ماذا سوف يكون الناتج طبعا اذا كانت الشيفرة صحيحة

قمت بالتعريف عن متغيرين باسم **num1,num2** اذا قام المترجم بحجز مكانين في الذاكرة ليس شرطا ان يكونا متجاورين كما اسلفنا وقلنا واحد للمتصرح الاول والثاني للثاني

السطر الخامس اعطيت قيمة للمتغير **num1** (10) اذا هنا تم تخزين الرقم 10 في المكان المحجوز للمتغير الذي صرحنا عنه وهو **num1** وقد حفظت به

السطر السادس قمنا بإعطاء قيمة جديدة للمتحول **num1**وهي قيمته المخزنة في الذاكرة مضافا الها (11) وعند ذالك قام المترجم باخذ قيمة المتحول المحجوز بالذاكرة وهو (10) (الى هنا لم يتم حذف هذه القيمةحتى وان استخدمت) واضاف اليها القيمة (11) وحسب التعليمة وضعها في المكان المخصص للمتغير (وبالتالي قام بتدمير القيمة القديمة ووضع بدلا عنها القيمة الجديدة)

في السطر السابع والثامن قام بنفس الافعال كما الخامس والسادس وانما للمتغير الثاني وفي موضع او مكان حجزه في الذاكرة

بالسطر التاسع قام واخذ قيمة المتغير **num1** وجمعها مع قيمة المتغير **num2**والتي كانت كما قلنا الاول اصبحت (21) الثاني اصبحت (10) وبعد الجمع اصبحو (31) وقام بوضعها في مكان المجوز للمتغير الاول **num1** بعد ان حذف او دمر ماكان القيمة الموجودة

السطر الاخير وهي كما قلنا تعليمة كتابة قيمة المتحول **num1**

لنتاكد ان البرنامج خالي من الاخطاء

نتاكد من الفواصل بنهاية كل تعليمة

تنويه24 عليك بوضع الفاصلة المنقوطة عند نهاية كل تعليمة واعتبرها قاااعدة طبعا الا في حالات قليلة لايتم وضع الفاصلة وسوف نشرحها كلما مررنا عليها

نتاكد ان قيمة المتحولات التي صرحنا بهم تتناسب مع نوعهم

اذاً كل شيئ صحيح نفذ المشروع ماهي النتيجة

تعالو لنتاكد ان قيمة المتحول الثاني **num2** لم يتم تدميرها من الذاكرة اكتب التعليمة التي بموجبها يتم طباعة قيمة المتحول وصرف المشروع انظر خرج البرنامج الاولى قيمة المتحول الاول والثانية قيمة الثاني

لنقارن مقارنة بسيطة بين اسلوب الباسكال وبين اسلوب البيسك

ففي البيسك يمكن اسناد اي نوع من القيم الى المتحولة مثال

شيفرة بيسك

x = -1 ;

x = 1100 ;

x = 3.4 ;

لماذا ....؟؟؟

لانه في لغة البيسك يهتم المترجم او المجمع بتخصيص مساحات تخزينية كافية لايستيعاب اي حجم من اي نمط كان طبعا من الاعداد

اما الباسكال مثلما راينا او السي++ كما سنرى الان يجب ان تصرح على نمط المتغير قبل استخدامه مما يسمح للمجمع القيام بتدقيق النوع والتاكد ان الامور تسير بالشكل الصحيح عندما يبدأ بتنفيذ البرنامج والاستخدام الخاطئ سوف يسبب خطأمصرف او تحذير يمكن تحليله وتصحيحه وبالتالي استئصال المشكلة قبل ظهورها

مثال سي++

```cpp
#include <iostream>
int main()
{
    int num1;
    int num2;
    int num3;

    num2 = (20) ;

    num2 = (10) ;

    num3 = num1 + num2;

    return 0;

}
```

من المه اخي ان تتذكر ان كل متغير او متحول او ماشئت ان تسميه يجب ايمون له اس **Name**

ونمط او نوع **Type** وحجم او المساحة التي سوف يتم حجزها له **Size** وقيمة **Value**كما سبق وذكرت .

طبعا هذا ليس كل شيئ عن المتغيرات فيوجد ايضا متغيرات السلاسل المحرفية او سلاسل الحروف سوف نتحدث عنها لاحقا

ولكن الان ارجو انه تكوَن لديكم اخواني مفهوما عاما عن مفهوم المتغير الية عمله متى يتم تدميره اليا دون ازعاج المبرمج ومتى يتم الاحتفاظ به وارجو ان تكون الامثلة وافية وكافية في ذالك

درسنا القادم عن العاملات بالباسكال والعمليات الحسابية وممكن ان ابدا بما يسمى عمليات المساواة والمقارنة

اعيد وانوه اي استفسار ممكن ان ترسلوه على العنوان

zahersaid937@hotmail.com