

# البرمجة بلغة باسكال

## Programming with Pascal

بقلم :  
وجدي عصام عبد الرحيم

## كلمه المؤلف:



بعد التطور الشديد في لغات البرمجه ، والاعداد الهائله من الطلبة التي تحاول ان تتعلم ولوشينا يسيرا لكي يسيروا في الدرب الاسلام لهم ، كان لا بد من لغة برمجه حتى ان لم تكن بمستوى التقنيات الجديده لكي يبدأ بها المبرمج الجديد في عالم البرمجه .

ومن هنا بزغت فكره انشاء كتاب يتحدث عن البرمج بلغه باسكال ، في الوقت الذي يندر ان نجد كتابا يتحدث عن هذه اللغه بشكل جيد .

في الحقيقه كثير من الكتب تتحدث عن لغة باسكال ، و لكن قليل منها يعلمك كيف تصبح مبرمجا، كثير من هذه الكتب مليء بأكواد سيئة جدا، سيئة من ناحية ان تصميمها سيء، و ليس من ناحية انها لا تعمل.

لذلك لا تتوقع الكثير من الامثله والتمارين المكرره كما في اكثر الكتب ، فأنا احب ان اشرح مفهوم البرمجه اي فهم الجمل والايعازات التي تتكون منها اللغه بشكل عام ، ثم شرح البرنامج بشكل خاص من حيث كتابه جمل سليمة خاليه من الاخطاء ، وما الخطوات البديله والاحتمالات الممكن استخدامها ، والاطاء الناتجه وكيفيه تفاديها وما الى ذلك.

لذلك ستجد الكثير من الشرح النظري، لكن صدقتي سوف يفيدك بشكل كبير في المستقبل في حال اردت كتابه برامج بصوره صحيحه وسليمه ، ايضا في حال اردت تغيير اللغه فسوف يفيدك كثيرا .

عموما اتمنى لكم اسعد الاوقات ، وبرمجه موفقه للجميع ، بإذن الله !!

المؤلف !!

وجدي عصام عبد الرحيم

## قبل قراءة هذا الكتاب



هذا الجزء الاول من الكتاب ، ايضا هي النسخة الاولى ، وسيتم تطويرها متى دعت الحاجة الى ذلك ، للحصول على آخر نسخة من هذا الكتاب قم بمراسلتي :  
• [romansy\\_766@hotmail.com](mailto:romansy_766@hotmail.com)

يمكنك طباعة هذا الكتاب للاستخدام الشخصي فقط، و يمنع استغلاله في أية أمور تجارية بدون الإذن الخطي من المؤلف.

في حالة بان وجدت جزء من هذا الكتاب غير واضحة أو بحاجة إلى شرح مفصل فبرجاء إخباري بذلك فوراً على بريدي الإلكتروني، حتى أتمكن من إضافة الأجزاء الناقصة أو توضيح الأجزاء الغير واضحة فوراً، طبعا في النسخة القادمة.

كل المعلومات التي في الكتاب تعبر عن رأي المؤلف وجدي عصام ، وهو غير مسؤول تماما عن أي استعمال غير شرعي لهذا الكتاب .

### للحصول على الدعم الفني ::

- قم بالتسجيل في المنتدى الخاص بالمؤلف :

[www.romansy2005.tk](http://www.romansy2005.tk)

- أو اضافة احد البريدين الاتيين ::

[romansy\\_766@hotmail.com](mailto:romansy_766@hotmail.com)

[wajdyessam@hotmail.com](mailto:wajdyessam@hotmail.com)

## المؤلف في سطور //

وجدي عصام ، اشرف على العديد من الساحات المهتمه بالامن والحمايه ، له العديد من المقالات والدروس في الكثير من المواقع والساحات العلميه ، حاليا هو مهتم بالبرمجه من الدرجه الاولى ، بدا حاليا في التعامل مع الانظمه المختلفه والمفتوحه المصدر ، ويحلم بالمزيد من التقدم في البرمجه وذلك بتعلم لغتي Delphi و Assembly ، ايضا هو مهتم ببرمجه المواقع وتطويرها ويحاول تعلم المزيد من اللغات كـ asp و html و xhtml و xml و css .

الاسم / وجدي عصام عبد الرحيم

اللقب / romansy

البلد / السودان

الايمل / [romansy\\_766@hotmail.com](mailto:romansy_766@hotmail.com)

***WhtEver You Learn... You learn it for your self!!***

# (( سبحانك لا علم لنا الا ما علمتنا انك انت العليم الحكيم ))



بفضل الله ورعايته ، تم الانتهاء من الجزئيه الاولى من كتاب ( **البرمجه بلغه باسكال** ) . هذه الجزئيه البسيطة والتي هي في الحقيقه عباره عن اهم الدروس والمواضيع التي تعتبر البدايه الحقيقيه لاي مبرمج يريد ان يضع رجليه بقوه في عالم البرمجه ، فعالم البرمجه اليوم ملئ باللغات التي اصبحت تتطور بشكل اشبه باليومي ، لذلك كان لا بد من لغه برمجه تشرح مفهوم البرمجه الحقيقي للمبرمج ، صحيح ان جميع لغات البرمجه كذلك ، لكن الفرق في صعوبه اللغات بعضها عن بعض . لذلك تم الاختيار على لغه باسكال نظرا لما تتمتع به من سهوله وسوف تلاحظونها خلال البدايه في اللغه ، هذا لا يعني ايضا انك ستتعلمها في حينها ، فالمبتدئ عموما ايا كانت لغه البرمجه سوف يواجه العديد من الصعوبات والمشاكل ، وسوف تبدأ بالتقلص تدريجيا مع كتابه العديد من البرمج وقراءه المزيد من الدروس والمواضيع .

اما بالنسبه للجزء الاخر ، سوف يكون في خلال الشهور القادمه ، نظرا لصعوبته من ناحيه ، ولقله المراجع والمصادر التي تتكلم عن بعض النقاط .

ايضا هذا الجزء يعتبر كافي للذين يريدون التنقل الى لغه اخرى فور ادراكهم للمنطق البرمجي والجزء الثاني يكون للاشخاص الذين قرروا ان يتعمقوا في هذه اللغه .

## تنويه /

بالنسبة لبعض الكلمات (في الاكواد ) سوف تلاحظ انها مكتوبه بالخط العريض (B) ، هذا يعني انها من كلمات لغه باسكال المحجوزه ، ايضا عند كتابة هذه الكلمات في برنامج الباسكال سوف تلاحظ ايضا انها بالخط العريض .

ايضا اثناء كتابة البرامج في هذا الباب سوف اقتصر فقط على كتابة ما بعد كلمه **begin** للبرنامج الرئيسي ولن نتطرق لكتابه البرنامج كاملا ( **الا في بعض البرامج**) نظرا لان كتابتها كامله سيؤدي الى زياده عشرات ان لم يكن مئات الصفحات بلا فائده .

ايضا سوف نستخدم كثيرا كلمه Block وهي تعني بدايه **begin** ونهايه **end** فأي برنامج باسكال يحتوي على الـ **block** الرئيسي وهو **begin** للبرنامج ومن ثم بعد كتابه البرنامج نضع نهايته وهي عباره عن **end.** ، ايضا قد تحتوي بعض جمل باسكال على **Block** خاص بها مثل جمل (**if** و **case** و **for .. الخ**) وسوف يتم ذكر هذه الجمل بالتفصيل في الدروس القادمه .

من افضل الطرق في التعلم ، هي كتابه المزيد من البرامج . والحذر من نسخ الاكواد ولصقها فإنها طريقه خاطئه جدا . عليك بقراءه الاكواد وفهمها ثم اعاده كتابتها مره اخرى في بيئه التطوير .

ايضا تم كتابه الاكواد في برنامج word لذلك عند محاوله نسخ الاكواد الى بيئه التطوير ، سوف تجدها غير مرتبه وبشكل تصعب قرائته . لذلك كما قلنا لا تقم بنسخ الاكواد.

## اليوم الاول

مرحبا بك في عالم باسكال ، اليوم ستبدأ رحلتك لتصبح مبرمجا خبيرا في لغه باسكال ياذن الله .

وسوف تعرف بعض المعلومات النظرية كنشأه اللغه ، ومن هو مكتشفها ومطورها . ايضا بعض المعلومات النظرية حول البرمجه الهيكلية .

وسوف تبدأ الخطوه الاولى في البرمجه ، وذلك من خلال تعلمك كيفية الوصول الى البرنامج ، وكيفية التعامل معه وترجمه البرامج ، وتنفيذها .

وكما يقال

مشوار الالف ميل يبدأ بخطوه .

بالتوفيق !

## تاريخ لغة باسكال :

تعود قصة الرياضي والفيلسوف الفرنسي بليز باسكال **Blaise Pascal** إلى منتصف القرن السادس عشر ميلادي عام 1642م حيث أن هذا الشاب والذي كان يبلغ الثامنة عشر ربيعا والذي كانت بواذر العبقرية والاختراع تبدو جلية على أعماله وحيث أنه كان شديد الإبداع في علم الرياضيات مولعا بالاكشافات العلمية وتطبيق النظريات الرياضية .

ففي نفس السنه اخترع أول حاسب نصف آلي وسمي باسمه ( **حاسب باسكال** 1642 ) ، وكان ذلك لسبب أساسي وهو مساعده أبيه الذي كان يعمل في مؤسسة الضرائب محصلا للفواتير والذي كان يقضي معظم لياليه مستخدما العد اليدوي في إحصاء وتدقيق حسابات المبالغ التي حصلها وقد كان يشكل هذا النوع البطيء من الحساب إرباكا لعائلته ، وبالتالي يأخذ منه الوقت الكبير .

يتكون حاسب باسكال الميكانيكي من مجموعه متتالية من الإطارات ( **الأقراص** ) كل واحد منها مرقمة من الصفر إلى التسعة ، هذه الإطارات مرتبه بحيث تقرأ الأرقام المسجلة عليها من اليسار إلى اليمين ويتم إدارتها يدويا عن طريق الذراع ، فعندما يتم أحد هذه الأقراص دورته من الصفر إلى التسع فإن نتوء الرقم 9 يدفع الطارة المجاورة له رقما واحدا وعند ذلك أي دورة الطارة الأولى تسعة مواقع متتالية ( **دوره كاملة** ) فإنها تدفع الطارة التالية لها من اليسار موقعا واحدا وهكذا حتى يتم تسجيل العدد. وبهذه الطريقة استطاع والد باسكال إجراء عمليات الجمع والطرح أما عمليات الضرب والقسمة فتتم بتكرار عمليات الجمع والطرح مرات متعددة.



## بدايه اللغه:

باسكال لغة برمجه انشئت بواسطة البروفيسور **Niklaus Wirth** في عام 1970 . كان اسمها سابقا ، نسبه الى عالم الرياضيات الفرنسي المشهور **Blaise Pascal** . لقد صنعت هذه اللغه لتعليم البرمجه ولكي تكون محل ثقته لدى المبرمجين . منذ ذلك الوقت قد اصبحت لغة البرمجه باسكال اكثر من انها فقط لغه اكاديميه و لكنها استخدمت تجارياً ايضاً ، وقد قرر منهج لغه باسكال في العديد من الجامعات في مختلف الدول ،

## المترجم :

يوجد العديد من المترجمات التي تعمل على باسكال ، ومن اشهرها الـ **Free Pascal** ايضاً هناك البورلاند **Borland Pascal** ، ايضاً هناك النسخه التريبو وهي التي سنتعمد عليها في هذا الكتاب **Turbo Pascal** ، وهناك ايضاً نسخه من شركه مايكروسوفت ، والعديد العديد .

الشيء الوحيد الذي تختلف فيه هذه المترجمات هي في كتابه بعض الدوال ، وطرق تنفيذها ، كل مترجم له مكتباته ودواله الخاصه . لكن الفكره البرمجييه هي نفسها في جميع المترجمات وليس لغه باسكال فقط وبالاصح لجميع لغات البرمجه الفكره البرمجييه واحده وان اختلفت طرق كتابه الكود بشكل لآخر ، لكن في النهايه الفكره واحد .

لذلك عند انتقال المبرمج المتقن للغته الى لغه لآخرى ، سيجد نفسه في وقت قصير اصبح مدرك للغه الجديده ليس لان اللغه الجديده سهله ، ولكن لان المبرمج اساسه متين و فاهم الفكره البرمجييه .

## البرامج

تستخدم كلمه برنامج **program** للتعبير عن معين ، الاول لوصف الاوامر المصدر **source code** التي يكتبها المبرمج ، والثاني لوصف البرنامج التنفيذي **executable software** بأكمله . هذا الاستخدام لكلمه برنامج قد ينتج عنه الكثير من الارتباك ، لذلك سنحاول التفرقه بين الاوامر المصدر من ناحيه ، والبرنامج التنفيذي من ناحيه اخرى .

البرنامج يمكن تعريفه بأنه مجموعه من الاوامر التي يكتبها المبرمج أو بأنه احد الاجزاء التنفيذيه من التطبيق .

يمكن تحويل الاوامر المصدر الى برنامج بطريقتين : الاولى هي ان يقوم المفسر بتحويل الاوامر المصدر الى اوامر يفهمها الحاسب بحيث يقوم بتنفيذها على الفور . والطريقه الثانيه هي ان يقوم المترجم بتحويل الاوامر المصدر الى برنامج ، والذي سيمكنك تشغيله في وقت لاحق . وبالرغم من سهوله استخدام المفسر ، فإن اغلب لغات البرمجه الجاده تستخدم المترجم ، لان الاوامر المترجمه يتم تشغيلها اسرع . وتندرج لغه باسكال ضمن اللغات المترجمه .

## الفرق بين المترجم والمفسر

يقوم المفسر بقراءه الاوامر المصدر ثم يترجم البرنامج ، بحيث يحول الاوامر التي كتبها المبرمج الى اجراءات مباشره ، اما المترجم فيقوم بترجمه الاوامر المصدر الى برنامج تنفيذي يمكن تشغيله في وقت لاحق ،

ايضا امكانيه توزيع البرنامج التنفيذي على اشخاص لا يملكون المترجم (اللغات المستخدمة للمترجم) ، اما بالنسبه للغات التي تستخدم المفسر فلا بد من وجود المفسر لتشغيل البرنامج .

## البرمجه الاجرائيه والهيكلية

حتى وقت قريب ، كان ينظر الى البرامج على انها سلسله من الاجراءات التي تستجيب للبيانات . والاجراء ما هو الا مجموعه من الاوامر المحدده التي يجري تنفيذها واحدا تلو الاخر . كان يتم فصل البيانات عن الاجراءات ، وكان جوهر البرمجه يكمن في معرفه الاجراءات التي استدعت اجراءات اخرى ، ومعرفه البيانات التي تم تغييرها .

وللتغلب على هذا الموقف الذي يؤدي الى احداث بعض الارتباك ، ثم انشاء البرمجه الهيكلية **structured programming** .

ان الفكره الرئيسيه المبنية عليها البرمجه الهيكلية تماثل في بساطتها فكره "فرق تسد" . يمكن تخيل البرنامج على انه مركب من مجموعه من مهام . واي مهمه اعقد من ان توصف بسهولة سيتم تقسيمها الى مجموعه من المهام الاصغر الى ان تصبح المهام صغيره بالدرجه الكافيه لفهمها واستيعابها بسهولة

على سبيل المثال ، ستجد ان حساب متوسط مرتبات الموظفين باحدى الشركات مهمه معقده ، ومع ذلك ، بمقدورك تقسيم هذه المهمه الى المهام التاليه :

1. معرفه مرتب كل موظف
2. حساب عدد الموظفين

3. جمع كافة المرتبات
4. تقسيم مجموع المرتبات على عدد الموظفين

من الممكن تقسيم مهمه جمع المرتبات (3) الى الخطوات التاليه :

1. احضار السجل الخاص بكل موظف
2. الوصول الى قيمه المرتب
3. اضافه المرتب الى اجمالي المرتبات
4. احضار سجل الموظف التالي

وهكذا ، يتم تقسيم المهام المقعده الى مهام فرعيه اصغر منها حتى تصبح المشكله يسيره .

## بيئه التطوير

كما ذكرنا سابقا ، تم اعتماد المترجم Turbo Pascal في هذا الكتاب ، والسبب في ذلك حيث يحتوي على عدد كبير من الاجراءات الجاهزه ، بخلاف بقية المترجمات التي تعمل على هذه اللغه ، هذا لا يعني ان البقيه غير مناسبه للعمل .

عند تنصيب برنامج Turbo Pascal ، عاده ياخذ المسار c:\tpw ، وتستطيع الوصول اليه من خلال قائمه ابدأ ، ثم الذهاب الى البرامج الملحقه ، ثم اختر tpw ، بعدها اختر TPW.EXE .



في الواجهه الاساسيه ، يمكنك كتابه الاكواد التي سوف تتعلمها لاحقا ، ايضا كما ذكرنا سابقا ، التعليقات سوف تكون باللون الازرق ، والكلمات المحجوزه سوف تكون بالخط العريض .

شريط العنوان ، يحتوي على مسار التطبيق الحالى .

شريط القوائم ، يحتوي على العديد من القوائم كما نلاحظ . وهي لن تختلف عن اي تطبيق اخر ( مثلا برنامج word ) . اما بالنسبه للقائمه compile ، فهي لترجمه البرنامج . والقائمه run فهي لتشغيل البرنامج ( اي تنفيذه ) .

شريط الادوات ، يحتوي على مجموعه من الادوات التي هي اختصار لبعض الاوامر الموجوده في شريط القوائم .

الشريط الذي يوجد في اسفل الصفحه (من الممكن ان نطلق عليه شريط **الحاله**) يحتوي على اربع خانات ، الاولى لمعرفه رقم الصف والعمود . الثانيه لمعرفه هل تم حفظ الملف بعد التعديل ام لا . الثالثه لتغيير طريقه

الكتابه ( تستطيع تغييرها من المفتاح insert الموجود في لوحة المفاتيح ) لكن لا انصح بتغييرها ابدا . الخانه الرابعه اهم خانه ، لانه يتم فيها عرض الاخطاء اثناء ترجمه البرنامج او اثناء تنفيذ البرنامج . وسوف يتم كتابه الخطأ في هذه الخانه .

## اليوم الثاني

تكمّن الصعوبه في تعلم موضوع معقد مثل البرمجہ  
في اعتماد كثير مما ستعلمه على كل شيء اخر يمكنك تعلمه .

في هذا اليوم سوف تتعلم التركيب الاساسي للبرنامج في باسكال ، ايضا  
سوف تتمكن من ادراك كيفيه كتابه برنامج يعتبر مقروء نوعا ما .

تتكون لغه باسكال من الوحدات ، والاجراءات ، والمتغيرات والثوابت  
في هذا اليوم ستتعلم اين يكون الاعلان عنهم

في هذا اليوم لا تعتقد انك سوف تتعلم الكثير ، لاننا كما ذكرنا انك  
سوف تتعلم المواضيع الجديده من خلال تقدمك في قراءه المزيد من  
المواضيع المتقدمه .

تعتبر لغة الباسكال من اللغات التركيبية structured ، هذا يعني انه اي شي يكتب فيها يجب ان يكون مرتب ومنظم ، لا الكثير من القفزات وجمل الـ Goto ، يعني لها صيغه او هيئه واحده ، حقا !!

مثال على هيئه برنامج بسيط ::

**Program** name (input, output);

**Uses**.....

**Var**

...

**Begin**

{Your program is here}

**End.**

## رأس البرنامج

السطر الاول هو لكتابه اسم للبرنامج ، اي اسم يمكنك وضعه كيفما تريد ، لكن يجب ان يكون له علاقه بالبرنامج الذي تكتبه ، فمثلا برنامج الـ حاسبه ، يصبح اسمه مثلا calculate ، ايضا هناك قواعد في كتابه اسماء البرامج فلا يمكن ان يحتوي على مسافات او يبدأ الاسم برقم ، ايضا في حاله ان اسم البرنامج يحتوي على كلمتين فمن الممكن ان تكتب متلاصقتين مع مراعاة تكبير الحرف الاول من كل كلمه ، فمثلا WajdyEssam ، او من الممكن وضع علامه الشرطه - للفصل بين الاسم الاول والثاني ، wajdy-essam .



اما الكلمتين input و output ، فهي لتوضيح هل البرنامج يحتوي على مدخلات (اي يقوم مستخدم البرنامج بادخال بيانات ) او مخرجات (اي تنطبع النتائج على الشاشة فور تنفيذ البرنامج ) .

فالبرامج التي تحتوي على مخرجات فقط ، مثلا طباعه جمله على الشاشة فور تنفيذ البرنامج فيكون output فقط ! اما اذا كان مدخلات فقط بدون مخرجات (مستحيله نوعا ما ، او بالاصح نادره) فتكون input فقط ، في حال البرنامج يحتوي على مدخلات ومخرجات (وهو الاغلب) فتكون الاثنتين معا ، اي input, output .

### Program example (output);

{مثال على برنامج يحتوي مخرجات فقط}

في الكثير من الاحيان ، يتم تجاهل الكلمتين input , output ولا يتم كتابتها على الاطلاق . ويكتفى باسم البرنامج فقط .

### Program test;

ايضا يمكن كتابه السطر الاول للبرنامج بهذا الشكل/

### Program clean (I ,o);

الحرفين I و o هما اختصار لكلمتي : input و output

كما ان رأس البرنامج غير ضروري لتنفيذ البرنامج ، اي انه من الممكن تجاهله تمام والبدء بالجمله التي تليه ، لكن هذا الخيار غير محبذ تماما .

شخصيا ، انا افضل (والعديد من المبرمجين) كتابه اسم البرنامج وتوضيح هل يحتوي البرنامج على مدخلات ومخرجات .

## قسم الاعلان عن الوحدات ( المكتبات المستخدمه )

الكلمه المحجوزه uses ، تستخدم للاعلان عن الوحدات التي سوف تستخدمها في البرنامج . هناك العديد من الوحدات الجاهزه التي تستطيع استخدامها في برنامجك ، او يمكنك انشاء وحداتك الخاصه (وسيتم شرحها بالتفصيل ، عندما يحين وقتها ) . لغه باسكال تستخدم اسم الوحدات unit ، بدل من اسم المكتبات library . ومن اشهر الوحدات المستخدمه في لغه باسكال : wincrt و windos و strings وغيرها .

// اذا للاعلان عن الوحدات //

Uses wincrt;

Uses wincrt, windos; { للاعلان عن الاثنتين }

بشكل مبسط ، الوحده unit هي عباره عن مجموعه من الاوامر والاجراءات التي تستخدمها في برنامجك ، مثلا عباره writeln و readln (سوف نتطرق لهما بعد قليل ) هي عباره عن اجراءات ، وتم تضمينها داخل الوحده . wincrt

اذا اي وحده هي عباره عن مجموعه من الاوامر والاجراءات ، وهذه الاوامر والاجراءات تكون داخل الوحده ، فعندما نستخدم اي امر في لغه باسكال ، يجب ان نكون اعلنا عن الوحده المتضمنه لهذا الامر .

## قسم الاعلان عن المتغيرات Var

الكلمه المحجوزه **Var** (عند كتابه الكلمات المحجوزه في المترجم ، فإنه يقوم بكتابتها بالخط العريض **Bold** ) تشير الى المتغيرات التي سوف تستخدمها في البرنامج ، المتغيرات هي مساحه في الذاكره يستخدمها باسكال لتخزين بعض المعلومات ، هذه المعلومات من الممكن ان تتغير على حسب الاوامر التي تعالج في لحظه تنفيذ البرنامج . في حاله تسميه المتغيرات يجب ان نتبع بعض القواعد ، فلا يبدأ برقم ، ولا يكون من الاسماء المحجوزه .. الخ ، وغيرها من الاشياء التي سوف نمر عليها لاحقا بالتفصيل .

### البرنامج الرئيسي

كل البرامج في باسكال ، تبدأ بالكلمه المحجوزه **begin** ، وتنتهي بالكلمه المحجوزه **end.** (لاحظ النقطه ، ومعناها ان نهايه البرنامج هنا ، واي شي يكتب بعد هذه النقطه ، لا يلتفت اليها المترجم ) .

في الكثير من البرامج ، سوف ترى ان هناك العديد من الجمل او الاوامر الخاصه بلغه باسكال ( مثل الجمله الشرطيه **if** ، وجمل التكرار **for** ) تحتاج الى بدايه **begin** ونهايه **end;** (لاحظ الفاصله المنقوطة **Semicolon ;** ) .

هذه الجمل ( التي تحتاج الى **begin** و **end;** خاصه بها ) تسمى جمل مركبه وتسمى ايضا **Block** .

يجب ان تلاحظ ، ان نهايه البرنامج دائما تكون بالجمله **end.** ، اما نهايه الـ **Block** (الجمله المركبه) يكون دائما بالجمله **end;** .

سوف نتكلم عن هذه النقاط بالتفصيل من خلال الدروس القادمه !!

## التعليقات Comments

هي عباره عن كلمات يضعها المبرمج ، لتوضيح وظيفه احد الاوامر ، او لكتابه بعض المعلومات عن البرنامج (مثلا متى بدأ في كتابه البرنامج ، متى انتهى أو اخر تطوير للبرنامج) .

وعندما يرى المترجم هذه التعليقات ، يقوم بتجاهلها والذهاب الى الجمل التي تليها ( اي انها لا تظهر وقت تنفيذ البرنامج ، بينما تظهر في الكود المصدر source code ) وهناك نوعين من التعليقات وهما :

(\* This is Comments \*) /1

{ This is another comments } /2

ولا يوجد فرق بينهما ، ويمكنك استخدام ما تشاء منهما .

والجمله {Your program is here} في المثال الاول ، هي عباره عن تعليق لا ينفذ في البرنامج .

// مثال على تعليق

**Program** wajdy (input, output);

**Uses** winCRT;

**Begin**

Writeln ('hello im wajdy essam'); {print my name}

**End.**

في اثناء كتابه التعليقات في برنامج باسكال ، سوف تكتب باللون الازرق الفاتح، في المثال السابق ، التعليق {print my name} ، ولن يظهر اثناء تنفيذ البرنامج .

اذا باختصار / التعليقات تستخدم لايضاح عمل بعض الدوال والاجراءات التي تكون معروفه لدى المبرمج وغير معروفه للبقية ، ايضا تستخدم لكتابه بعض المعلومات حول البرنامج (كوقت انشاءه وتحديثه) او معلومات حول المبرمج نفسه (كالاسم وعنوان البريد الالكتروني) .

تكون التعليقات في اي مكان في برنامج باسكال ، اي مكان على الاطلاق ، لكن يفضل ان تكتب التعليقات في بدايه البرنامج ( هذا اذا كانت معلومات حول وظيفه البرنامج ، او معلومات حول المبرمج ) ، او تكتب بجانب الاوامر الغير معروفه ، كما في المثال السابق ( باعتبار ان الامر writeln غير معروف ) .

## ما قبل البرنامج الرئيسي

(اي ما قبل الـ begin والـ end) هذا المكان يخصص للاعلان عن جميع الاشياء التي سوف تستخدمها في برنامجك ، اقصد بجميع الاشياء /

راس البرنامج (كما ذكرنا سابقا) يعلن عنه بالكلمه المحجوزه **program**  
الوحدات المستخدمه (كما ذكرنا) يعلن عنها بالكلمه المحجوزه **uses**  
العنوانين المستخدمه في البرنامج يعلن عنها بالكلمه المحجوزه **label**  
الجملة المعرفه من قبل المستخدم يعلن عنها بالكلمه المحجوزه **type**

المتغيرات (كما ذكرنا) المستخدمه يعلن عنها بالكلمه المحجوزه **Var**  
الثوابت المستخدمه في البرنامج يعلن عنها بالكلمه المحجوزه **const**  
الاجراءات المستخدمه في البرنامج يعلن عنها بالكلمه المحجوزه **Procedure**  
الاقتارات المستخدمه في البرنامج يعلن عنها بالكلمه المحجوزه **Function**

وسيتم ذكر جميع هذه الاشياء ، بشيء من التفصيل عندما يحين وقتها .

كما لاحظت ( اظن ذلك ) ان اي سطر في لغه باسكال ينتهي بالفاصله المنقطه semicolon وهي ( ; ) . معنى هذه الفاصله ان الجملة ( او الامر ) انتهى .

## استخدام المسافات والـ Tab لتحسين شكل البرنامج :

من الميزات التي تتميز بها لغه باسكال ، المقروئيه Readability !! ماذا تعني ؟ تعني ان البرنامج في باسكال مكتوب بشكل منظم وسهل يسهل التتبع والقراءه بكل يسر ، هذه الميزه جعلتها من اللغات الاساسيه التي تدرس في اغلب المعاهد والجامعات ، بالاضافه الى هذه الميزه في باسكال ، يمكنك ان تحسن من شكل البرنامج اكثر فأكثر . كما سيأتي .

من اهم الاشياء لدى المبرمج عند كتابة برنامج ما ، هو ان يكون البرنامج واضح ومقروء للجميع (المبرمجين) ، حتى للمبرمج نفسه ، في حاله اراد تصحيح خطأ ما ، او اضافه قطعه من الكود !! في هذه الحاله يجب ان يكون البرنامج مرتب وايضا يحتوي على تعليقات في حال استخدام دوال غير مألوفه وقد تم ذكر كيفية استخدام التعليقات .

نأتي الى تحسين شكل البرنامج ، وذلك باستخدام مفتاح الـ Tab وهو موجود في الجهة اليسرى من الكيبورد .  
انظر المثال الاتي (لا تتوقع ان تفهم البرنامج الان ، فقط خذ فكره عامه)  
المثال الاول:

```
Program clean (I , o);  
Uses wincrt;  
Var  
X: integer;  
  Begin  
    Read(x);  
    If x>10 then writeln (' bigger then 10 ' )  
    Else  
      Writeln (' less then 10 ');  
  End.
```

المثال الثاني:

```
Program clean (I , o);  
Uses wincrt;  
Var  
X: integer;  
  Begin  
    Read(x);  
    If x>10 then writeln (' bigger then 10 ' )  
    Else  
      Writeln (' less then 10 ');  
  End.
```

تمعن النظر فيهما ، الان ما هو الاسهل قراءه في نظرك .....؟؟

اظن الاول اسهل بكثير ، اليس كذلك !! لانه تم استخدام الـ Tab ، اما عن طريقه استخدامه فهي سهله جدا ، اذا كنت تريد ان تحرك النص الى اليمين كل ما عليك هو ضغط مفتاح Tab وبعد كتابه الجمله وضغط enter سوف ينتقل المؤشر الى اسفل الجمله السابقه مباشره (اي التي تم قبل كتابتها استخدام مفتاح الـ Tab ) .

اظن العمليه نظريا صعبه نوعا ما ، لكن عند محاول استخدامها سوف تجدها في غايه السهوله ، وبكتابه بعض البرامج العديده سوف تلاحظ انك اصبحت قادرا على استخدامها بمهاره كبيره ، لا تقلق اذا لم تدرك استخدامها الان .

في حال تسميه البرنامج يراعى تسميته باسم يدل على فعله او وظيفته ولا يمكن كتابه اسم للبرنامج ويكون هو نفسه اسم لمتغير ، ايضا لا يمكن استخدام المسافات في كتابه اسم البرنامج ، والاسماء الطويله غير محبذه بتاتا في كتابه اسم برنامج ، كن حريص !!

في الصفحات السابقه استخدمنا العديد من الجمل او المصطلحات التي اظن انك لم تفهمها او تجيدها ، مع كل هذا انت في الطريق الصحيح تماما ، هذه هي الحال في جميع لغات البرمجه ، في البدايه ستجد نفسك غير مستوعب أي شيء لكنك من خلال كتابة العديد من البرامج وقراءه المزيد المزيد من الدروس ستجد نفسك مبرمج ، مبتدئ 😊



## اليوم الثالث

### المتغيرات والثوابت

تحتاج البرامج الى اتباع طريقه لتخزين البيانات التي تستخدمها تقدم الثوابت والمتغيرات العديد من الطرق لتمثيل هذه البيانات ومعالجتها ،

#### ستتعلم اليوم

- ❖ كيفية طباعه جملة على الشاشة
- ❖ كيفية الاعلان عن المتغيرات والثوابت وتعريفهما
- ❖ كيفية تعيين القيم للمتغيرات ومعالجه هذه القيم
- ❖ كيفية كتابه قيمه احد المتغيرات على الشاشة

## جمله الطباعه في الشاشه :

في لغه باسكال هناك جملتين للطباعه على الشاشه وهم:

write /1

writeln /2

**الفرق بينهما :** هو ان write يطبع الجمله على الشاشه ثم يكون المؤشر في نفس السطر .

اما في جمله writeln فان الجمله تتطبع ايضا على الشاشه لكن المؤشر ينتقل الى السطر التالي .

## هناك ثلاث انواع من الطباعه :

- طباعه نص على الشاشه
- طباعه قيمه متغير على الشاشه
- طباعه نص وقيمه متغير على الشاشه

## النوع الاول

لطباعة اي نص على الشاشه يجب ان نقوم بكتابه امر الطباعه ( **write** او **writeln** ) ثم نقوم بفتح قوس واغلاقه ايضا ونكتب بداخله الجمله المراد طباعتها ، ايضا يجب ان نحيطها بقوس صغير ( ' ) quote من الجانبين ( يتم كتابته عن طريق مفتاح حرف (ط) باللغه الانجليزيه )

مثلا:

1/ Write (' wajdy ');

2/ Writeln ('Welcome to My Program!! ');

في المثال الاول سوف يطبع في الشاشة ( عند تنفيذ البرنامج ) كلمه **wajdy** ويكون المؤشر في نفس السطر .

في المثال الثاني سوف يطبع على الشاشة **Welcome to My Program !!** وسوف ينتقل المؤشر الى السطر التالي ( بسبب استخدام **writeln** ) .

مثال (لم اكتب البرنامج كامل) /

**Begin**

Writeln ('I learn Pascal');

Writeln ('Hi, there !');

**End.**

الناتج من هذا البرنامج هو:

**I learn Pascal**

**Hi, there**

اما في حاله كتابه الكود السابق بجمله **write** ،،

**begin**

Write('I learn Pascal');

Write ('Hi, there !');

**end.**

الناتج من هذا البرنامج هو:

I learn Pascal Hi, there

في حاله اننا نريد ان نترك مسافه سطر بين الجمله الاولى والجمله الثانيه  
سوف تكتب : writeln;

// مثال

لطباعه جملته وترك سطر فارغ ثم طباعه جملته اخرى سوف يكون شكل  
البرنامج كالاتي ::

**Begin**

Writeln ('My Name is wajdy essam');

Writeln;

Write ('do you understand the lesson');

**End.**

المخرجات سوف تكون كالاتي :

My name is wajdy essam

Do you understand the lesson?

## Hello World اول برنامج لك بلغه باسكال

هناك تقليد متبع في كتب البرمجه ، وهو أن تبدأ هذه الكتب دائماً بإنشاء برنامج باسم Hello يقوم بعرض العبارة الترحيبية Hello World على الشاشة . وهذا التقليد القديم جدير بأن يؤخذ به في هذا الكتاب .

**Program Hello (output);**

**Uses wincrt;**

**Begin**

Writeln ('Hello World ! ');

**End.**

اكتب البرنامج داخل بيئه التطوير ، وقم بترجمته ثم بتشغيله . ستجد الشاشة تعرض : **Hello World !**

اذا تم ذلك فتهانينا لك ! فقد نجحت في كتابه اول برنامج لك في باسكال ، قد يبدوا البرنامج بسيطا للغاية ، لكن عليك ان تدرك ان جميع المبرمجين المحترفين بدؤوا بهذا البرنامج .

## شرح البرنامج //

السطر الاول ، هو لكتابه اسم البرنامج Hello ، وحيث ان البرنامج يحتوي على مخرجات فقط ، نجد ان الجملة التوضيحية الخاصه بمعرفه هل البرنامج له مدخلات او مخرجات ، هي مخرجات output فقط .

السطر الثاني ، هي الاعلان عن الوحده المستخدمه في البرنامج ، ونظرا لان الامر write و writeln هي من الاوامر الموجوده داخل الوحده wincrt فيجب الاعلان عنها ( اذا اي برنامج يحتوي على مخرجات يجب ان يتم الاعلان عن هذه الوحده ) .

السطر الثالث ، هو لبدايه البرنامج الرئيسي begin .

السطر الرابع ، هو لطباعه جمله ! Hello World على الشاشة ، وتم استخدام الامر writeln ، اي سينتقل المؤشر الى السطر التالي بعد تنفيذ الامر .

السطر الاخير ، هو لانهاء البرنامج end. ( لا تنسى النقطه )  
النوع الثاني والثالث من انواع الطباعه ، سوف يتم شرحهم بعد شرح المتغيرات .

## ما هو المتغير

في اغلب لغات البرمجه المتغير variable هو مكان لتخزين المعلومات ، المتغير هو موقع بذاكره جهازك حيث يمكنك تخزين قيمه بداخله ثم استعادته هذه القيمه منه فيما بعد .

من الممكن تصور ذاكره جهازك على هيئه سلسله من الخانات ، وكل خانه تمثل واحده من عده خانات متراصه ، وجميع الخانات ( او مواقع الذاكره ) يتم ترقيمها تسلسليا .

تعرف هذه الارقام بأنها عناوين الذاكره ، يقوم المتغير بحجز خانه او اكثر بحيث تخزن فيها احدى القيم .

سيمثل اسم المتغير ( وليكن myvariable ) بطاقه عنونه ملصقه على احدى هذه الخانات بحيث تستطيع الوصول اليه سريعا بدون معرفه عنوانه في الذاكره ،

يعرض الشكل التالي تخطيطا يمثل هذه الفكرة . وكما يتضح من الشكل ، يبدأ المتغير من عنوان الذاكرة 102 . واستنادا الى حجم المتغير myvariable فقد يأخذ واحد أو اكثر من عناوين الذاكرة .

اسم المتغير

*Myvariable*

الذاكرة



العنوان

100

101

102

103

104

## حجز الذاكرة //

عند تعريفك لاحد المتغيرات في باسكال ، يجب ان تخبر المترجم بنوع هذا المتغير ، هل هو عدد صحيح Integer ام حرف character ، ام غير ذلك ؟ هذه المعلومة تخبر المترجم بالمساحة التي سيحجزها وكذلك نوع البيانات المراد تخزينها بالمتغير

كل خانه تشغل واحد بايت . اذا كان حجم المتغير الذي تنشئه اربعة بايت ، فهذا معناه انه سيحتاج الى اربعة بايت من الذاكرة . او اربعة خانات . يخبر نوع المتغير (مثل العدد الصحيح integer ) المترجم بمقدار الذاكرة (عدد الخانات ) التي سيحجزها المتغير .

هناك العديد من المتغيرات المختلفه في لغة باسكال ايضا بأبعاد مختلفه ،، وحجوم مختلفه ايضا !! هذه هي اشهرها ::

اسم المتغير	الم (من) — (الى) — دى	الحجم	النوع
Shortint	-128 الى +127	1 بايت	integer
Byte	0 الى 255	1 بايت	integer
Integer	-32768 الى +32767	2 بايت	integer
Word	0 الى 65535	2 بايت	integer
Longint	-2146473648 الى +2146473647	4 بايت	integer
Real	-???????? الى +????????	6 بايت	fractional
String	بحدود 255 حرف	255 بايت	غير رقمى
Char	حرف واحد فقط	1 بايت	غير رقمى
Boolean	True - False	1 بايت	غير رقمى

يوجد ايضا المزيد ، لكن هذا كل ما نحتاجه الان .

## الاعلان عن المتغيرات

كما رأينا سابقاً (اليوم الثاني) في الشكل العام لبرنامج باسكال ، رأينا عبارة **var** وهنا في هذا الجزء سوف نعلن عن جميع المتغيرات التي نحتاجها في البرنامج (كل برنامج له متغيرات على حسب المطلوب من البرنامج) .

## الشكل العام للاعلان عن المتغيرات :

; نوعه : اسم المتغير  
variablename : type



## Variable name

هنا تضع اي اسم كيفما تريد ، لكنك لا تستطيع وضع مسافات داخل الاسم. او البدايه برقم في احد اسماء المتغيرات ، ومن بين اسماء المتغيرات التي تستطيع استخدامها x و myage وهكذا ، اسماء المتغيرات الجيده هي التي تعرفك بالعناصر التي يتم استخدام المتغيرات لها ، استخدام الاسماء الجيده يسهل من فهم واستيعاب خطوات سير البرنامج .

// مثال

**Var**

X:integer;

Y:string;

A,b:char; {للاعلان عن متغيرين من نوع واحد}

عند اعلانك عن متغير ، يتم تخصيص (حجز) الذاكره لهذا المتغير ، قيمه هذا المتغير ستكون اي قيمه تصادف وجودها بالذاكره في هذا الوقت ، ستعرف بعد قليل كيفيه تعيين قيمه جديده لهذه الذاكره .

**وكقاعده عامه في التسميه :**

- حاول تجنب الاسماء المبهمه مثل j5d8g
- حاول استخدام المتغيرات ذات الحرف الواحد x و i
- حاول استخدام الاسماء المعبره مثل count و MyAge

## تعيين القيم لمتغيراتك

لاسناد (تعيين) المتغير الى قيمه معينه، يجب كتابته بالشكل الاتي :

```
var_name:= value;
```

وهذا التعيين يجب ان يكون داخل بدايه البرنامج ،أي بعد الـ begin للبرنامج الرئيسي .

// مثال

**Var**

X: integer;

**Begin**

X: = 10;

Writeln(x);

**End.**

المخرجات سوف تكون عباره عن قيمه X وتساوي 10 .

## شرح البرنامج :

في البدايه (بعد كتابه اسم البرنامج والوحده لم اكتبهما بغرض الاختصار والسرعه ،قم بكتابتهما بنفسك في حال اردت تنفيذ البرنامج) نقوم بالاعلان عن المتغيرات في الجزء المخصص لها ، وفي المثال قمنا بتعريف X من النوع integer أي كعدد صحيح ، ثم بدأنا البرنامج وذلك من خلال العبارة begin ، بعدها قمنا بتعيين (اسناد) الرقم 10 الى المتغير X ، ثم قمنا بطباعه

المتغير x عن طريق جملة الطباعة writeln(x) لاحظ اننا لم نستخدم الاقواس الصغيره ' ' (وهذا هو النوع الثاني من انواع الطباعة).

لماذا لم نستخدمها ??? سؤال منطقي !!

عندما نريد ان نطبع قيمه متغير ما فنكتبها بدون اقواس (النوع الثاني).  
اما في حاله طباعه الجملة كما هي نستخدمها بالاقواس (النوع الاول).

مثال (باعتبار ان x من النوع string ) //

```
Write (' I Learn Pascal ');
```

```
X:= 'O.k.';
```

```
Writeln(x);
```

// الناتج من البرنامج ( المخرجات )

I Learn Pascal O.K

شرح البرنامج

السطر الاول هو لطباعه الجملة I Learn Pascal وحيث ان امر الطباعة هو write فإن المؤشر سوف يكون في نفس السطر.  
السطر الثاني ، تم تعيين الكلمه o.k في المتغير x .  
السطر الثالث ، هو طباعه قيمه المتغير x (وهي o.k) .

مثال اخر //

قرر ما هي قيمه x باعتبار انها من النوع الصحيح integer ?

```
X: = 10;  
X: = 2;  
Write(x);
```

## المخرجات هي / 2

### شرح البرنامج

في السطر الاول ، تم تعيين قيمه X وهي 10 ، ثم في السطر الثاني تم تعيين قيمه جديده للـ X (اي تم مسح القديمه وتعيين اخرى بدلها وهي 2) وفي السطر الاخير تم طباعه قيمه X ، وهي 2.

### النوع الثالث من انواع الطباعه

وهو طباعه نص وقيمته لمتغير ، وتكون بالشكل الاتي //

1. WriteLn ('The total is ' , x);
2. WriteLn ('my name is: ' , n, 'my family name is:', f);

في المثال الاول باعتبار ان قيمه x تساوي 100  
سوف تكون المخرجات كالآتي /

**The total is 100**

في المثال الثاني باعتبار ان قيمه n تساوي wajdy وقيمته f تساوي essam  
سوف تكون المخرجات كالآتي /

**My name is: wajdy my family name is: essam**

وهذا هو النوع الثالث من انواع الطباعه ، والغرض منه الاختصار فقط ! فبدل ان نكتب جمله طباعه نص (النوع الاول) ثم نكتب جمله طباعه قيمه متغير (النوع الثاني) ، تصبح الاثنان في جمله طباعه واحده ، ويتم التفريق بينهما (اي بين النص وقيمته المتغير) بواسطه فاصله (,).

## جمله القراءه على الشاشة :

كثير من الاحيان ، تحتاج الى ان يقوم المستخدم بادخال قيمه ما في اثناء تنفيذ البرنامج ، واعتمادا على هذه القيمه سوف تكون هناك نتائج .

وفي باسكال هناك جملتين للقراءه الا وهي /

read /1

readln /2

## الفرق بينهما :

في العبارة الاولى read ، يتم قراءه المتغيرات المدخلة من المستخدم ، ويتوقف المؤشر عند اخر متغير تمت قرائته . وفي الاخر ينزل الى السطر التالي .

اريد ان انوه انه في عمليات الادخال " لا يهم اذا ادخلت العدد الاول ثم مسافه ثم العدد الثاني ، او ادخلت العدد الاول ثم انتر ثم العدد الثاني " .

مثال على الادخال:

## Begin

```
Write('Enter you number: ');  
Readln(x);  
Writeln ('You Number is: ', x);
```

## End.

## شرح البرنامج

بدأ البرنامج بطباعه نص على الشاشة وهو Enter you number: الان وفي نفس السطر (بسبب ان جمله الطباعه هي write) سوف نقوم بقراءه المتغير X (قم بتعريفه بنفسك!) . وينتقل المؤشر الى السطر التالي. اخيرا سوف نقوم بطباعه نص + قيمه متغير (وهو النوع الثالث من انواع الطباعه) .

## المثال الثاني

**Program** example (input, output);

**Uses** wincrt;

**Var**

X: integer;

S: string;

**Begin**

```
Write ('Enter your Name: ');
```

```
Readln(s);
```

```
Write ('Enter your Age: ');
```

```
Readln(x);
```

```
Writeln ('Your Name is: ',s,' your age is: ',x);
```

**End.**

## المخرجات من البرنامج

Enter your Name: wajdy

Enter your Age: 20

Your Name is: wajdy your age is: 20

## شرح البرنامج

السطرين الاولين ، معروفين الاول لاسم البرنامج والثاني للوحده المستخدمه ( نظرا لان هناك مخرجات (عباره writeln) فيجب ان نستخدم وحده ( winCRT ) .

السطر الثالث ، هو للاعلان عن المتغيرات . وقد تم الاعلان المتغير x من النوع الصحيح integer ، والمتغير s من النوع السلسله النصيه string .

السطر الرابع لبدايه البرنامج begin ، السطر الخامس لطباعه الجمله Enter your Name: على الشاشة ويكون المؤشر في نفس السطر ، والسطر السادس لقراءه قيمه s والانتقال الى السطر التالي ، السطر السابع لطباعه الجمله Enter your Age: على الشاشة ويكون المؤشر في نفس السطر ، السطر الثامن لقراءه قيمه x وينتقل المؤشر الى السطر التالي .

السطر التاسع هو لطباعه النص Your Name is وقيمته المتغير s وطباعه النص your age is: وقيمته المتغير x .

## الثوابت Constants

في حياتنا الواقعيه كثيرا منا نجد العديد من الاشياء الثابته التي لا تتغير قيمتها ابدا ، مثلا/

$$Pi = 3.1415926513 , e = 2.7182818284529$$

هذه الارقام من الصعب حفظها ، للبعض منا على الاقل ، لذلك باسكال تقدم اداه جيده للثوابت ، ايضا توجد بعض الدوال المعرفه مسبقا فيها مثلا **pi**.

### الاعلان عن الثوابت

يكون عاده فوق الاعلان عن المتغيرات ، كما يأخذ الشكل الاتي /

قيمته : اسم الثابت  
Constname: Value;

مثال على الاعلان

### Const

Myconst = 1234 ;

اي ان الثابت myconst ، تكون قيمته 1234 ، ولا تتغير اثناء تنفيذ البرنامج ، كما ان تسميه الثوابت تخضع لنفس شروط تسميه المتغيرات ، راجع درس المتغيرات .

مثال بسيط لاستخدام الثوابت //



**Program** example (input, output);

**Uses** wincrt;

**Const**

Mc=50;

**Var**

X: integer;

**Begin**

X: = 2;

Write(x\*mc);

**End.**

أرأيت ، كم هو بسيط الاستخدام ، قم بتجربه البرنامج ، ما هي المخرجات؟؟  
المخرجات هي عبارته عن قيمه ضرب  $50*2$  وتساوي **100** .

لكن تذكر كلمتي هذه ، الثوابت غير قابله للتغير ابدا ابدا الا في حاله واحده فقط وهي عندما يكون الاعلان عن الثابت بهذا الشكل /

**Const**

Myconst: mytype =the value;

// مثال

**Const**

Mc: integer = 40;

/ ايضا

## Const

```
Aa: integer=2, 3;
```

هذا المثال خاطئ ؟ لماذا

لان الثابت Aa من النوع الصحيح integer ولا يمكن ان تكون قيمته من النوع الحقيقي real أي 2.3

واضح ، الان كيف يمكن ان نغير قيمته اثناء البرنامج .

**مثال //**

## Const

```
X:word=10;
```

## Var

```
Z:integer;
```

## Begin

```
  Z:= 2;
```

```
  X:= 20;
```

```
  Writeln(x*z);
```

## End.

سهل ، اليس كذلك ، كما في المتغيرات العاديه .

## اليوم الرابع

### التعبيرات والعبارات

البرنامج في حقيقته عبارته عن مجموعه من الاوامر التي يجري تنفيذها بالتسلسل ، وممكن القوه في البرنامج يتمثل في امكانيه تنفيذ مجموعه او اخرى من الاوامر استنادا الى صحه او خطأ شرط معين .

ستتعلم اليوم :

- ما هي العبارات والتعبيرات
- ما هي الكتل
- العمليات (حسابيه ومنطقيه) في لغه باسكال

## العبارات

تتحكم العبارات statement في تسلسل التنفيذ ، او تقوم بتقييم التعبير ، او لا تفعل اي شي . تنتهي كاقه العبارات بفاصله منقوطة ( ; ) . ومن اشهر العبارات ، عباره التعيين (الاسناد) ، كما تم ذكره سابقا .

**X:= a+b;**

وعلى عكس ما هو متعارف عليه في علم الجبر ، لاتعني هذه العباره ان x يساوي مجموع a و b . وانما تقرأ " قم بتعيين مجموع a و b للمتغير x " وبالرغم من قيام هذه العباره بمهمتين ، فانها تعامل كعباره واحده ولذلك لا تشتمل الا على فاصله منقوطة واحده . معامل التعيين assignment يعمل على تعيين ما هو موجود بالجانب الايمن لعلامه لعلامه التعيين (:=) لما هو موجود بالجانب الايسر .

## الكتله والعبارات المركبه block

اي مكان تستطيع فيه وضع عباره مفرده ، سيمكنك كذلك وضع عباره مركبه فيه ، والتي تسمى بالكتله (او العباره المركبه) ، وتكون عن طريق كتابه begin ثم وضع العبارات التي تريدها ، بعد ذلك قم باغلاق الكتله بالكلمه end; (لاحظ وجود الفاصله مع كلمه end) .

## التعبيرات

اي شي ينتج عنه احدى القيم يوصف بأنه تعبير expression . والتعبير هو ما يعود باحدى القيم . فالعباره 2+3 تعود بالقيمه 5 ، جميع التعبيرات هي في نفس الوقت عبارات .

## العمليات الرياضيه والمنطقيه في باسكال:

### • العمليات الرياضيه :

وهي تتكون من عمليه الجمع والضرب والطرح والقسمه (عدد صحيح ، عدد حقيقي ) وباقي القسمه ، وتفصيل ذلك فيما يلي :

العمليه	شكلها
الجمع	+
الطرح	-
الضرب	*
قسمه عدد صحيح	div
قسمه عدد حقيقي	/
باقي القسمه	mod

### • العمليات المنطقيه :

وتتكون من علامه اكبر من واصغر من ويساوي .. الخ ، وتفصيل ذلك كما يلي :

العمليه	العلامه
اكبر	>
اكبر من او يساوي	>=
اصغر	<
اصغر من او يساوي	<=
يساوي	=
لا يساوي	<>

## اولا : العمليات الرياضيه /

كما اظن ، ان اغلب البشر يكونوا على علم بهذه العمليات . فعمليه الجمع تكون بالاشاره + والطرح - والضرب \* والقسمه هناك نوعين ، الاول هو قسمه الاعداد الصحيح integer (اي يتم تعريفه من النوع الصحيح في قسم تعريف المتغيرات) وتكون بالكلمه div ، الثاني هو قسمه الاعداد الحقيقيه وتكون باشاره القسمه العاديه / ، باقي القسمه يكون بالكلمه mod .

• الاعداد الصحيح integer هي الاعداد الصحيحه التي لا تحتوي على كسور مثلا / 5 و 10 و -52 (اعداد سالبه ، موجبه ، ولا تحتوي على كسور).

• الاعداد الحقيقيه real هي الاعداد التي تحتوي على كسور (ايضا موجبه او سالبه ) مثلا / 2.5 و -9.2 و 10 (الاعداد الصحيحه من ضمن الاعداد الحقيقيه ، اي ان الحقيقيه اوسع واشمل ، اذن جميع الاعداد الصحيحه هي حقيقيه ، والعكس غير صحيح ) .

## قسمه الاعداد ، والباقي من القسمه

عندما نقسم 24 على 4 ،  $24 \div 4$  ، يكون الناتج هو 5 ، والمتبقى من القسمه هو 1 .

وللحصول على المتبقى ، يكون بهذا الشكل :  $24 \bmod 4$  .

## اسبقية المعاملات

في اي عميله حسابيه ، سيقوم المترجم بالبدء في الحساب على حسب اسبقية كل عمليه ، لذلك يجب التنبيه اليها في حال اردت نتائج سليمة . وهي كما يلي /

- الضرب
- القسمة
- الجمع
- الطرح

في حال وجود معاملين حسابيين لهما نفس الاسبقية ، فسوف يجري تنفيذهما من اليسار الى اليمين . ففي العبارة

$$X:= 5+3+8*9+6*4;$$

يجري عمليه الضرب اولاً، من اليسار الى اليمين . وعلى اساس ان  $9*8=72$  وان  $6*4=24$  ، فإن العبارة ستصبح :

$$X:= 5+3+72+24;$$

والان ستجري عمليه الجمع من اليسار الى اليمين  $5+3=8$  و  $8+72=80$  و  $80+24=104$  .

في بعض الاحيان ، يتوجب عليك استخدام الاقواس لتغيير ترتيب الاسبقية ، حيث ان الاقواس تأخذ درجه اسبقية اعلى من جميع المعاملات الحسابيه .

## تعشيش الاقواس

ماذا تعني ؟ في الكثير من الاحيان سنجد ان التعبير اصبح مقعد الى درجه مخيفه ، لذلك نلجأ الى تخفيف ذلك التعبير ، عن طريق اضافه متغيرات لكل عباره في هذا التعبير المقعد .

Total:= ( (old+new)/2) \* (water+phone);

تتم قراءه هذا التعبير المقعد من الداخل الى الخارج .

الان لو حاولنا تبسيطه باستخدام التعشيش

All:= old+new;

Average := all/2;

House:= water\*phone;

Total:=average \* house ;

ارأيت ، صحيح انه يأخذ وقتا اطول في الكتابه ، كما يستخدم متغيرات اكثر ، لكنه اسهل كثيرا في الاستيعاب .

## ثانيا: العمليات المنطقية /

كما هو واضح في الجدول السابق ، اكبر ، اصغر ، يساوي ... الخ ، وهي ترجع بالقيمه true اذا كانت العلميه صحيحه  $3 < 6$  ، وترجع بالقيمه false اذا كانت العلميه غير صحيحه  $5 = 1$  .



اليوم الخامس

الجمل الشرطيه

في حياتنا العاديه ، كثيرا ما تواجهنا العديد من المواقف التي نحتاج فيها الى اتخاذ احد الشروط

فمثلا، اذا فهمت الدرس فسوف انتقل الى الدرس التالي والا سوف اعيد قرائته مره اخرى !

في لغه باسكال توجد جملتين للشرط /

● جمله **if**

● جمله **case**

وسوف نتطرق لهما بالتفصيل .

## اولا : جمله if الشكل العام لها

```
If condition then  
Begin  
:  
End;
```

او اذا قمنا باضافه كلمه والا :

```
If condition then  
Begin  
:  
End {No Semicolon ; }  
Else  
Begin  
:  
End;
```

حسنا ، كيف يفهم المترجم ذلك ؟ اولا يقوم بالتحقق من الشرط condition ، اذا كان الشرط متحقق فانه يقوم بتنفيذ ما بين (الـ Block) Begin و end; وما بعد جمله والا else فانه يقوم بتجاهلها (هذا في حاله الشرط متحقق) .

اما اذا لم يكن متحقق فانه يذهب الى جمله else ويقوم بتنفيذ ما الـ Block الخاص بالجمله else .

يجدر الاشاره هنا انه في اغلب جمل لغه باسكال ( **if** و **for** و **case** و...الخ ) انه اذا كان بعد الجمله (مثلا جمله **if** ) عباره واحده فانه غير مطلوب ان نضع Block أي **begin** و **end;** .

اما اذا كان بعد الجمله **if** اكثر من جمله مراد تنفيذها في حاله تحقق الشرط فانه يجب ان نضع الـ **block** .

//مثال

**Begin**

Read(x);

**If** x<10 **then**

**Begin**

Sum:=x+20;

Write(sum);

**End;**

Write('done');

**End.**

**شرح البرنامج :**

بعد تعريف المتغيرات sum , x في بدايه البرنامج (قم بتعريفهما بنفسك ) ، يقوم البرنامج في البدايه بقراءه المتغير x ، ثم يقرر اذا كانت X هل هي اصغر من 10 ؟؟ اذا نعم (اي تحقق الشرط ) فانه يقوم بتنفيذ ما بين الـ block (أي ما بين **begin** و **end;** ) .

اما اذا كان الشرط غير متحقق (مثلا قيمه الـ  $x=20$ ) فإنه يقوم بتجاهل الـ block ويقوم بطباعه الجملة التي تليها **done**.

### ملاحظه:

في حال تحقق الشرط (مثلا  $x=5$ ) فإنه سيتم تنفيذ الـ block وبعدها سيقوم بطباعه الجمل التي تليه **done**.

### //مثال

```
Program wajdy (input,output);  
Uses wincrt;  
Var  
I:integer;  
Begin  
  Write('Please enter any number: ');  
  Readln(i);  
  If i <=10 then  
    Begin  
      Writeln('hello world');  
      Writeln('wajdy essam');  
      Writeln('O.K');  
    End;  
  Writeln('do you anderstand ?? ');  
End.
```

## شرح البرنامج

في البدايه قمنا بطباعه الجملة **Please enter any number** على الشاشة.  
مالفائده منها؟؟

أي برنامج يجب ان يحتوي على توضيح من المبرمج ، فلربما قام احد المستخدمين بتجربه البرنامج ، فيجب ان توضح للمستخدم ما يجب ان يدخله في البدايه ، هل هو عبارته عن عدد صحيح عن اسم او عن عدد حقيقي او حرف ، وهكذا .

اذا الغرض منها التوضيح لمستخدم البرنامج ، فأنت لا تصنع البرامج لتستخدمها لنفسك ، فيجب عليك في اي برنامج توضيح ما يجب ان يدخله المستخدم قبل اي عمليه ادخال .

في العبارة الثانيه قمنا بقراءه قيمه المتغير  $i$  ، ومن ثم بدأت الجملة الشرطيه اذا كان العدد المدخل  $i$  اصغر من او يساوي العشره فإنه يقوم بتنفيذ الـ `block` . اما اذا لم يتحقق الشرط فإنه يقوم بتجاهل الـ `block` .

## المخرجات

• في حاله  $i=5$  /

```
Please enter any number: 5
hello world
wajdy essam
O.K
do you anderstand ??
```

• في حالة  $i=20$  /

Please enter any number: 20  
do you understand ??

مثال اخر //

```
Begin  
  Read(x);  
  If x>0 then  
    Begin  
      Writeln('You Enterd 0');  
      Writeln('o.k');  
    End  
  Else  
    Begin  
      Writeln('you enterd not 0 ');  
      Writeln('o.k');  
    End;  
  Writeln('done');  
End.
```

المخرجات

اذا كان المدخل 0 فإن المخرجات هي :

You Enterd 0  
o.k  
done

اما اذا كانت غير الصفر فإن المخرجات هي:

**You Entered not 0**

**o.k**

**Done**

لاحظ اختفاء علامه ; قبل جمله **else** ، وهكذا في جميع جمل **if**. اذا كانت تحتوي على جمله **else** فإن الجملة التي تكون قبلها لا تحتوي على semicolon لأنها تعتبر جمله واحده من البدايه .

### جمله if المتداخلة Nested if

ما هو الحال مع جمله if بداخلها جمله if اخرى ، كيف يمكن ان يكون ؟

//مثال

**Begin**

**If x<10 then**

**Begin**

Writeln(' x is less than 10 ');

**If x>3 then write('x is more than 3');**

**End;**

**End.**

نعم نعم ، كم هو جميل ، ايضا من الممكن ان تحتوي if الثانيه (في البرنامج السابق) على اكثر من امر او عباره وفي هذه الحالة يجب ان توضع داخل block اخر (أي **begin** و **end;** ) .

## Combining the conditions ربط الشرط

تقدم باسكال اربعة طرق لربط الشروط بعضها ببعض وهي :  
not, And , or, xor

### جمله and و :

معناها انه لا يتم تنفيذ الشرط او block الا اذا كان الشرطين متحققين

### جمله or او :

معناها انه لا يتم تنفيذ الشرط او block الا اذا كان احد الشرطين (او كلاهما) متحقق

### جمله xor :

معناها انه لا يتم تنفيذ الشرط او block الا اذا كان هناك شرط واحد فقط متحقق

### جمله النفي not :

أي انها تنفي الشرط ، يعني اذا كان الشرط true فإنه يصبح false والعكس أيضا



**Begin**

Read(x,y);

**If** (x<10) **and** (y>5) **then**

**Begin**

Writeln('o.k');

Writeln('helow');

**End;**

**End.**

### شرح البرنامج

في البدايه ( بعد تعريف المتغيرات x و y من النوع integer ، قم بتعريفهما بنفسك ) قمنا بقراءه قيمه كل من x و y . ومن ثم بدأ الشرط ، اذا كان X اصغر من 10 و y اكبر من 5 فإنه سيقوم بتنفيذ الـ block .  
بمعنى اذا كان X=8 و y=5 فإنه سيقوم بطباعه الاتي :

o.k

Helow

اما اذا كانت x=12 و y=6 فإنه لن ينفذ أي شي .

قم بتجربه البرنامج بكل جمل الربط السابقه ، ولاحظ الاختلاف.

ايضا لاحظ وضع الاقواس في كل شرط ، اي ان كل عمليه ربط يجب ان توضع كل شرط في قوس .

الان ، كيف هو الحال مع ثلاثه شروط ؟؟

سيكون شكل التنفيذ كالآتي :

**If ( (x<10) and (y>5) ) or (k>3) then .....**

نعم نعم ،

أي انه في البدايه سيقوم بتنفيذ ما بين القوس الكبير ( (x<10) and (y>5) ) ثم تبدأ عمليه **or** مع المتغير الثالث (k>3) .

مثلا: اذا تم ادخال القيم الآتية **x=7 y=6 k=4**

الان هل x اصغر من 10 و y اكبر من 5 ؟؟  
نعم

الان هل k اكبر من 3 ؟؟؟ نعم

إذا تحقق الشرط ،،،

حسنا ، اذا كانت قيمه k=2 هل سيتنفذ الشرط ؟؟؟  
نعم

لماذا ؟؟؟

لان الشرط الاول تحقق ( (x<10) and (y>5) ) والعمليه **or** تنتفذ اذا كان هناك احد الشروط متحقق .

## ثانيا: جمله Of .. case :

في الكثير من الاحيان تحتاج الى تكرار العديد من الشروط في برنامج واحد يعني العديد من جمله **if**، ولكن في هذه الحالة يكون البرنامج طويل ومعقد ايضا، بالنسبه للبعض على الاقل ومن هنا جائت عباره **case** كبديل لجمله **if**

لنقل ان لدينا برنامج لدرجات الطلاب ،، ويكون المطلوب كالآتي :

اذا كانت درجه الطالب مابين 90-100 يطبع A

اذا كانت درجه الطالب مابين 80-89 يطبع B

اذا كانت درجه الطالب مابين 70-79 يطبع C

اذا كانت درجه الطالب مابين 60-69 يطبع E

اذا كانت درجه الطالب مابين 50-59 يطبع D

اذا كانت درجه الطالب مابين 0-49 يطبع F

كيف يمكن حل هذا البرنامج ؟؟

اولا: بجمله **if** :

**Begin**

Read(x);

**If (x>=90) and (x<=100) then writeln('A');**

**If (x>=80) and (x<=89) then writeln('B');**

**If (x>=70) and (x<=79) then writeln('C');**

**If (x>=60) and (x<=69) then writeln('D');**

**If (x>=50) and (x<=59) then writeln('E');**

**If (x>=0) and (x<=49) then writeln('F');**

**End.**

## شرح البرنامج

في البرنامج اعلاه ، قمنا بحل البرنامج بجمله **if** ، في البدايه (بعد تعريف المتغير **x** من النوع **integer**) قمنا بقراءه قيمه المتغير (**x**) ثم في بدأنا في جمل الشرط .

اذا كانت الدرجه اكبر من او تساوي 90 واصغر من او تساوي 100 فاطبع  
A

اذا كانت الدرجه اكبر من او تساوي 80 واصغر من او تساوي 89 فاطبع B  
وهكذا في جميع الجمل الباقيه .

## جمله in

تستخدم جمله in في الجمله الشرطيه لتحديد المدى ، دعنا نرى كيفيه استخدامها في المثال السابق .  
فمثلا العبارة الاولى:

```
If (x>=90) and (x<=100) then writeln('A');
```

سوف تصبح بعد استخدام الجمله in (وهي لتحديد المدى)

```
If x in [90 .. 100] then writeln('A');
```

ارأيت ، كم هو سهل ، يتم تحديد المدى بكتابه العنصر الاول والاخير وبينهم نقطتان فقط ( .. ) ، لاحظ نقطتان فقط !

ايضا لو كان هناك اعداد معينه في حال استخدام الجمله in سوف تصبح بالشكل الاتي :

```
If z in [5,9,6,8,66] then writeln('O.k');
```

او حروف ايضا :

**If ch in ['C','B','U'] then writeln('CPU');**

مع الاخذ بالاعتبار ان المتغير ch سوف يكون من النوع char او string .

نعود الى المثال الاول (درجات الطالب) ، حيث (في المثال الاول) قمنا بكتابه البرنامج بجمله **if** ولكن ما رأيك لو قمنا بكتابه بـ **if** المتداخله (أي جمله **if** داخل جمله **if** اخرى).

**Begin**

Read(x);

**If x >= 90 then mark:= 'A'**

**Else**

**If x >= 80 then mark:='B'**

**Else**

**If x >=70 then mark:= 'C'**

**Else**

**If x >=60 then mark:='D'**

**Else**

**If x >= 50 then mark:='E'**

**Else**

**If x >=0 then mark:='F';**

**Writeln(mark);**

**End.**

## شرح البرنامج

في البدايه قمنا بتعريف متغيرين الاول x من النوع integer والثاني mark من النوع char .  
بعدها قمنا بقراءه قيمه المتغير x ، ثم بدأت الجمل الشرطيه .

**If x >= 90 then mark := 'A'**

معنى هذا السطر اذا كانت قيمه x اكبر من او تساوي 90 فقم بتخزين A في المتغير mark ( لاحظ العلامه ' على جانبي A لانها من النوع char ) ثم اضفنا جملة والا else (أي اصغر من او تساوي 89) وقمنا بكتابه الشرط الثاني ، واضافه else ايضا ، وهكذا في جميع الجمل الباقيه .

فإذا لم تتحقق الجمله الاولى ، سيذهب الى الجمله الثانيه ، فإذا لم تتحقق سيذهب الى الجمله الثانيه وهكذا ، الى ان يتحقق الشرط ، وينتقل الى جملة الطباعه .

لاحظ اختفاء علامه ; (semicolon) في جميع الجمل التي يأتي بعدها جملة else الا الشرط الاخير بسبب ان ما بعده ليس جملة else .  
وفي الاخير قمنا بطباعه قيمه المتغير mark .

الان نعود الى جملة case ، الشكل العام للجمله :

**Case variable of**

:

**End;**

لاحظ ان جملة case لا تحتوي على begin ولكن تحتوي على end; .

## الان لحل برنامج الدرجات بجملة case

### Begin

Read(x);

### Case x of

90..100 : writeln('A');

80..89 : writeln('B');

70..79 : writeln('C');

60..69 : writeln('D');

50..59 : writeln('E');

0..49 : writeln('F');

End;

### End.

في البدايه (بعد تعريف المتغير x من النوع الصحيح ، حيث جملة case تقبل جميع انواع المتغيرات ، ماعدا المتغير من النوع الحقيقي real ) قمنا بقراءه المتغير x . فإذا كانت قيمه x (درجه الطالب) تتحصر بين 90 الى 100 فيتم طباعه A على الشاشة ، واذا كانت تتحصر بين 80 الى 89 فيتم طباعه B على الشاشة . وهكذا

حسنا ، اذا كانت قيمه x تساوي -10 او 200 ما هي المخرجات ؟ لن تكون هناك اي مخرجات ، لذلك يجب وضع جملة else ونضع بها رساله خطأ error ، حتى تكون اي قيمه لا تقع في المجال 0..100 لها رساله خطأ . (لا تنسى وضع النقطتان ، اثناء كتابه المدى . في حاله عدم وجود مدى يمكن وضع العناصر مع الفصل بينهم بفاصله ) .

## مثال /

إذا كانت النتيجة a و b فيتم طباعه very good ، وإذا كانت النتيجة c و d فيطبوع good ، وإذا كانت الدرجة e فيطبوع bad .

### Begin

Read(x);

### Case x of

A,b:writeln(‘Very Good’);

C,d:writeln(‘Good’);

E:writeln(‘bad’);

### Else

Writeln(‘Please enter any value between (a,b,c,d,e) ‘);

End;

### End.

اي في حال كتابه مدى معين يكون ذلك عن طريق كتابه العنصر الاول في المدى ثم النقطتان ( .. ) ، ثم نكتب العنصر الاخير في المدى .

اما اذا اردت كتابه عناصر مختلفه ، فتم كتابه العناصر والفصل بينهم بفاصله كما في المثال السابق .

ايضا لو كان هناك اكثر من امر في الشرط الاول أي عندما تكون الدرجة ما بين 90 الى 100 يجب ان يكون هناك مثلا جملتين طباعه على الشاشة كيف يمكن فعل ذلك ؟..



ببساطه نحتاج الى عباره begin نكتب بدخلها جميع الاوامر المراد ادخالها  
ثم انهاؤها بعباره end;

مثال/

**Case x of**

90..100:

**Begin**

Writeln('Your good man');

Writeln(' Your mark is A');

Writeln('Have anice day');

**End;**

80..89:

**Begin**

Writeln('Thank you very mach');

Writeln('Your mark is B');

**End;**

0..49:

**Begin**

Writeln('Im sorry');

Writeln('your maek is F');

**End;**

**Else**

Writeln('wrong');

**End;**

**End.**

في المثال السابق في حال كانت قيمه  $x$  تساوي 94 ، سوف تكون المخرجات  
عبارة عن :

Your good man

Your mark is A

Have anice day

وماذا اذا كانت قيمه  $x$  تساوي -60 او 70 ،،،؟؟ لاحظ اننا استخدمنا عبارة  
else

معناها أي قيمه لـ  $x$  غير المذكوره سوف يكون المخرجات wrong

## اليوم السادس

### الحلقات والتكرارات

في العديد من الاحيان ، سوف نريد تكرار جملة (مثلا تكرار طباعه اسمك على الشاشة) ومن غير المنطقي ان يتم كتابه جملة الطباعه لعدد كبير مثلا 100 مره .

او ان يكون عدد التنفيذ مجهول لدى المبرمج ، لارتباط هذا العدد بشرط او شروط معينه !

لذلك قدمت باسكال ثلاثه انواع من التكرار كل منها له وظيفه او غرض معين وهم :

- For ..... To .... do
- While ..... Do
- Repeat ..... until

وقبل البدايه في هذه الجمل ، يجب ان نعرف نشأه التكرار وبدايته (جملة

goto) .

## جذور التكرار (Goto)

في البدايات الاولى لعلم الحاسب، كانت البرامج في غاية القصر والتعقيد في نفس الوقت . وكانت التكرارات مكونه من عنوان ، وبعض العبارات ، وقفزه.

في لغه باسكال ، العنوان **label** عباره عن اسم متبوع بنقطتين ( : ) . يتم وضع العنوان على يسار اي عباره ، وتؤدي القفزه من خلال كتابه **goto** متبوعه باسم العنوان .

ويكون العنوان **label name** ( **name** ترمز الى اسم العنوان ) بعد الاعلان عن الوحدات ( **اي بعد uses** ) ، وفي البرنامج نضع النقطه التي صنعناها **name** متبوعه بنقطتين ( : ) ، وعندما نريد الذهاب اليها نضع جمله **goto name**.

مثال //

```
Program ex (input, output);  
Uses wincrt;  
Label wajdy;  
Var  
X: integer;  
Begin  
    Read(x) ;
```

```
If x >= 10 then exit;
if x < 10 then goto wajdy ;
Wajdy:    writeln ('Less Than 10');
End.
```

## الشرح

في المثال اعلاه ، تم الاعلان عن العنوان **label** باسم **wajdy** ، وفي بدايه البرنامج ، وضعنا الشرط التالي : اذا كانت قيمه  $x$  اكبر من او تساوي 10 فقم بالخروج من البرنامج **exit** . اما اذا كانت قيمه  $x$  اقل من 10 فتوجه الى العنوان **wajdy** ، في السطر التالي ، العنوان متبوعا بنقطتين راسيتين ويحتوي على جملة طباعه **less than 10** .

## السبب في تجنب استخدام **goto**

هناك الكثير من حملات الهجوم على عبارات تكرار **Goto** في الاونه الاخيره ، وهي تستحق ذلك بكل تأكيد . يمكن ان تؤدي عباره **goto** الى انتقال التنفيذ الى اي موقع في الاوامر التي تكتبها ، سواء للامام او للخلف

لقد ادى الاستخدام غير المقيد لعبارت **Goto** الى وجود برامج متشابهه يستحيل قراءتها تعرف باسم " **الاوامر الاسباجيتي** " ( **نظرا لتشابكها** ) ونتيجته لهذا ظن امضى معلمو علوم الكمبيوتر في الاونه الاخيره في صب درس واحد في رؤوس تلاميذهم ، الا وهو " **لاستخدم عبارات **goto** مهما كانت الاسباب او الظروف** " .

ولتجنب استخدام **Goto** ظهرت اوامر تكرر اكثر تطورا واحكاما ، من بينها **For** و **while** و **repeat** . استخدام هذه الاوامر جعل البرامج اكثر سهوله من حيث استيعابها ، وادى الى تجنب استخدام **Goto** بصفه عامه ،

## النوع الاول :: حلقه for ::

الشكل العام //

**For** variable := fromwhat **to** what **do**

**Begin**

:

:

**End;**

في البدايه قبل كتابه جمله **for** ، يجب ان يكون هناك متغير معرف على حسب التكرار المطلوب . بمعنى اذا التكرار من 1 الى 10 يكون المتغير من النوع الصحيح integer ، اما اذا كان التكرار من الحرف a الى الحرف z فيكون المتغير من النوع char ، وهكذا .

بعد تعريف المتغير variable نقوم بوضع علامه اسناد ( := ) ومن ثم نكتب القيمه الابتدائيه للتكرار fromwhat ثم نضع كلمه الى أي **to** ثم نضع القيمه النهائيه للمتغير what ونضع بعدها جمله افعل أي **do** .

يجب التنبه اذا كان المراد من التكرار جملة واحده ، فإنه من غير الضروري وضع Block لحلقه for . اما اذا كان التكرار لأكثر من جملة فيجب ان نضع block لحلقه For . (كما في جملة if و case ) وستبين ذلك من خلال الامثله التاليه .

مثلا لطباعه الاعداد من 1 الى 10 :

```
Program loop1 (input,output);
```

```
Uses wincrt;
```

```
Var I:integer;
```

```
Begin
```

```
For I := 1 to 10 do
```

```
Writeln(i);
```

```
End.
```

المخرجات من البرنامج السابق سوف تكون عبارته عن الاعداد من 1 الى 10 وكل عدد مطبوع في سطر لأننا استخدمنا جملة writeln في حال اردنا طباعتهم في نفس السطر نستخدم جملة write فقط .

**كيفية عمل الحلقة ،، (في المثال السابق)**

بعد بدايه البرنامج begin ، سوف يرى المترجم الحلقة من 1 الى 10 ، هذا يعني ان الجملة التي بعد do سوف تكرر 10 مرات ، اي عندما يأخذ القيمه 1 ثم ينفذ الجملة ، وعندما يأخذ 2 سوف ينفذها ايضا وهكذا . دعنا نأخذها بشيء من التفصيل (اي كما يراها المترجم ) .

اولا يأخذ المتغير I القيمة الابتدائية له (وهي 1) . ثم ينتقل الى الجملة التاليه وهي طباعه قيمه المتغير I ووضع المؤشر في السطر التالي ، فيقوم بطباعه الرقم 1 وانزال المؤشر الى السطر التالي (بسبب writeln) .  
الان يأخذ المتغير I القيمة الثانيه له (وهي 2) ثم ينتقل الى الجملة التاليه (فيطبوع الرقم 2 وينتقل المؤشر لاسفل) .  
وهكذا الى ان تنتهي الحلقة بتنفيذ اخر قيمه لها وهي طباعه الرقم 10 .

ايضا العداد من الممكن ان يكون تنازلي أي من 10 الى 1 ويكون شكله كالتالي:

```
For i:= 10 downto 1 do  
WriteLn(i);
```

في هذه الحاله (السابقه) سوف يكون الناتج من البرنامج عبارته عن الاعداد مطبوعه (كل واحده على سطر) من 10 الى 1 .

في حاله اردنا عداد لطباعه الاحرف (يجب ان يكون المتغير من النوع char).

```
For i:= 'a' to 'z' do  
Write(I, ' ');
```

معنى هذه الجملة write(I, ' '); هي ان يطبع القيمة الاولى للعداد (للحلقه) وهي a ومن ثم وضع مسافه ، الان طباعه القيمة الثانيه للعداد وهي b وترك



فراغ ايضا ، وهكذا الى ان يصل الى z (لاتنسى كل هذه العمليات في نفس السطر) .

```
write(I, ' '); // لو اردنا ان نطبع مسافتين
```

```
write(i:2); // من الممكن كتابتها بهذا الشكل
```

ومعناها لطباعه القيمه الاولى في فراغين (يعني فراغ ثم القيمه الاولى 1)

ثم طباعه القيمه الثانيه في فراغين (فراغ ثم القيمه الثانيه وهي 2)

الى ان يصل الى القيمه العاشره ويطبعا في فراغين ايضا (الفراغ الاول 1

والفارغ الثاني 0)

```
1 2 3 4 5 6 7 8 9 10 // سوف يكون شكل الحلقة كالتالي
```

حسنا لو كانت الجملة `writeln('hallow':10);` سوف يطبع الجملة كلها من

خلال العشره مسافات ، يعني سوف يكون هناك عشره مسافات منها اربعة

فارغه وسته للجملة `hallow` .

## يجب ملاحظه الامور التاليه في حلقه for :

1. تقوم الحلقة بتنفيذ الحلقه من اول قيمه الى اخر قيمه لها

2. في حاله كانت القيمه الابتدائيه هي نفسها القيمه النهائيه ، فالحلقه

سوف تنفذ مره واحده .

```
For i:=1 to 1 do  
Writeln('wajdy');
```

ناتج هذه الجملة هي طباعه كلمه **wajdy** مره واحد فقط

3. يجب ان يكون نوع المتغير في الحلقه هو نفسه نوع القيم الابتدائيه والنهائيه فمثلا باعتبار ان المتغير I من النوع integer (قرر هل الامثله التاليه صحيحه ام لا)

- ✚ For i:= 1 to 's' do
- ✚ For i:='a' to 'z' do
- ✚ For i:=10 to 3 do
- ✚ For i:=5 downto 10 do

المثال الاول : خاطئ

لان القيمه النهائيه s ليست من نوع متغير العداد

المثال الثاني: خاطئ

لان القيمه الابتدائيه والنهائيه ليست من نوع متغير العداد

المثال الثالث: خاطئ

لان الحلقه تنازليه ويجب استخدام جملة dwonto بدل جملة to

المثال الرابع: خاطئ

لان الحلقه تصاعديه ويجب استخدام جملة to بدل جملة dwonto

4. جملة for تكرر فقط الجملة التي تليها ، مثلا

```
For i:=1 to 10 do  
Writeln('wajdy');  
Writeln('essam');
```

الناتج من البرنامج هو عشره كلمات wajdy (كل منها في سطر) وفي الاخير كلمه essam . ان التكرار يأخذ الجملة التي تليه فقط ، فإذا اردنا ان يأخذ اكثر من جملة يجب ان نحصرها في block أي ان نضع begin ثم الجمل المراد تكرارها، ثم end; للتكرار، ثم end. للبرنامج الرئيسي .

**الان ، دعنا نأخذ مثال //**

لو طلب من تكرار الجملة التاليه 10 مرات  
الجملة هي : في السطر الاول hallow والسطر الثاني world ؟

```
Program loop2 (input,output);  
Uses wincrt;  
Var I: integer;  
Begin  
For i:=1 to 10 do  
Begin  
Writeln('hallow');  
Writeln('world');  
End;  
End.
```

ارأيت ، الان دعنا نشرح كيفية عمل هذه الحلقة (المركبه) أي تحتوي على block خاص بها ، أي لديها begin و end; خاصه بها.

### كل هذه الاسماء لها نفس المعنى //

حلقة مركبه ، block ، begin و end; ، كتله

في البدايه يأخذ المتغير I القيمه الابتدائيه له وهي 1 . وسوف ينتقل الى الجملة التاليه ، سوف يجد هناك begin و end; !! هنا سيفهم المترجم ان مابين begin والـ end; سوف ينفذ كل مره عندما تتغير قيمه المتغير I .

بمعنى عندما يأخذ المتغير I القيمه الاولى 1 سوف يطبع الكلمه hallow ثم ينتقل للسطر التالي ويطبع world ثم ينتقل المؤشر للسطر التالي . كل هذه العمليات في القيمه الاولى ( 1 ) فقط !!

الان يأخذ المتغير I القيمه الثانيه ، وسيقوم بتنفيذ الـ block (طباعه الاوامر الموجوده فيه).

ثم سيأخذ المتغير I القيمه الثالثه ، ويقوم بتنفيذ الـ block . وهكذا حتى يقوم بتنفيذ القيمه النهائيه ( 10 ) .

## النوع الثاني :: حلقه while ::

// الشكل العام //

```
While condition do  
begin  
:  
:  
End;
```

### كيفية عملها:

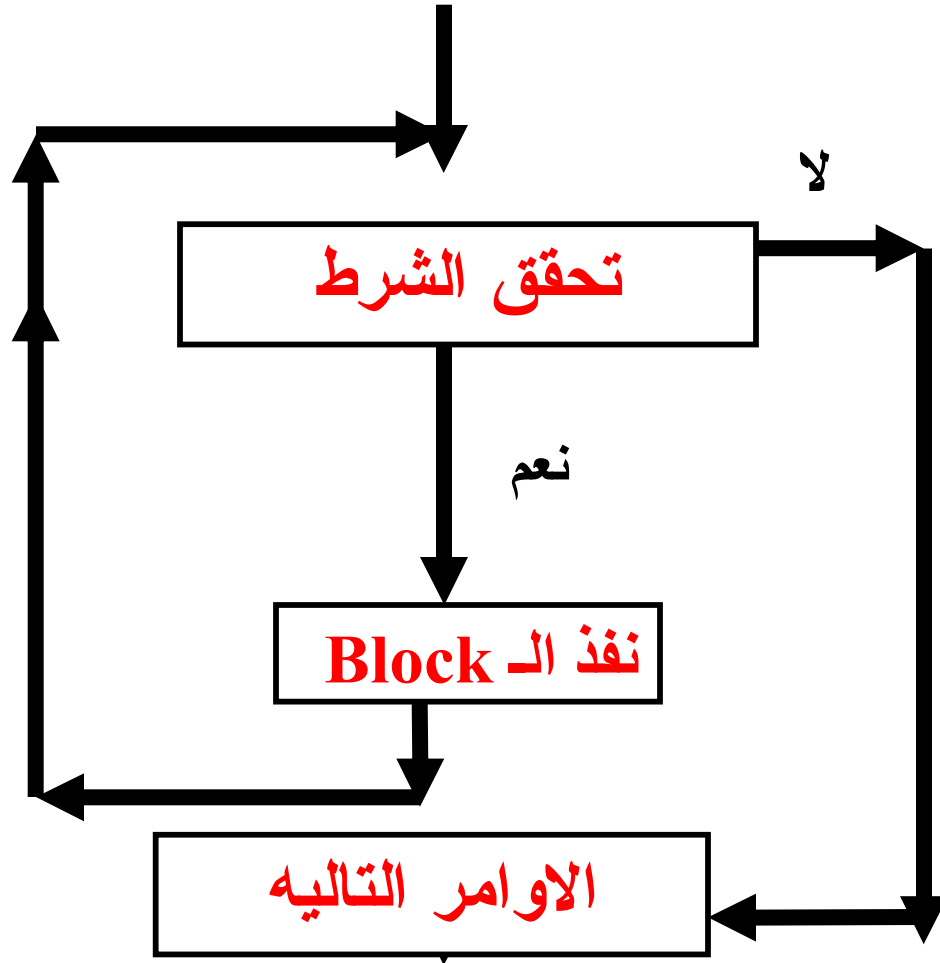
في البدايه يتم التحقق من الشرط condition ، هل هو متحقق ام لا ؟  
اذا نعم ، فإن التكرار سوف يأخذ القيمه الاولى وينفذ الاوامر التي توجد في  
الـ block حتى يصل للـ end; .

فإنه يتحقق مره اخرى هل تحقق الشرط ام لا ..؟

اذا نعم فإنه يقوم مره ثانيه بتنفيذ الـ block . الى ان لا يتحقق الشرط .

اذا لم يتحقق فانه يقوم بالخروج من هذه الحلقه ، وتنفيذ الاوامر بعد جمله  
end; اذا كان هناك جمل او اوامر .

// كما في الشكل الاتي //



مثال // لطباعه الاعداد من 1 الى 5 بحلقه while //

**Program** loop3 (input,output);

Uses wincrt;

**Var**

I:integer;

**Begin**

I:=1;

**While** i<6 **do**

**Begin**

Writeln(i);

I: =i+1;

**End;**

**End.**

## شرح البرنامج :

في البدايه (بعد تعريف المتغير I من النوع الصحيح integer). قمنا بوضع قيمه ابتدائيه للمتغير I (يجب وضعها حتى يعرف المترجم البدايه ، والا سيضع الافتراضيه له وهي دائما 0 ) ايضا في الجمله الثالثه من جمل التكرار repeat يجب وضع قيمه ابتدائيه للتكرار والا ستكون الافتراضيه وهي 0 .

الان بدأنا التكرار ، قمنا بكتابه الشرط  $i < 6$  ، معناه اذا كانت قيمه I اصغر من 6 سيتم الدخول الى داخل الـ Block اما اذا كانت تساوي 6 او اكبر فإنه يقوم بالخروج منه وعدم تنفيذ شي (في الـ block طبعا )

اذا تحقق الشرط (اي I اصغر من 6 ) سوف تتم طباعه المتغير I . واضافه الجمله  $i := i + 1$  ، يعني I القديمه (1) + الرقم 1 ، وقمنا بتخزين الناتج (2) في متغير I .

الان يعود الى الشرط مره اخرى هل I اللي هي 2 اصغر من 6 ؟ نعم ، اذا يقوم مره اخرى بتنفيذ الـ block (وسوف تكون قيمه الـ I هذه المره تساوي 3)

وهكذا سوف تكرر العمليه الى ان تكون قيمه المتغير I تساوي 6 في هذه  
الحاله سوف يرجع الى الشرط هل I (6) اصغر من 6 .... الجواب لا طبعا  
اذا سوف يقوم بالخروج من التكرار .

## ملاحظات ::

يجب عليك في اغلب الاحيان ان تعطي قيمه ما للمتغير الذي لديك قبل  
جمله **while** (وايضا **repeat**) ، اذا لم تريد فإن المترجم سوف يعطيها  
قيمتها الافتراضيه وهي 0 .  
في المثال السابق احذف هذه الجمله نهائيا ( **i:=i+1** ) ، وقم بتنفيذ البرنامج  
ماذا تلاحظ ؟ سوف تجد ان الناتج هو عباره عن الاعداد من 0 الى 5  
مطبوعه كل منها في سطر .

اذا ما العمل لكي يصبح البرنامج يطبع الاعداد من 1 الى 5 وليس من 0 ؟  
علينا بتغيير مواقع الجمل التي بداخل الـ block أي نضع الجمله الاولى  
**i:=i+1** ثم نضع الجمله الثانيه هي **writeln(i)** .  
ايضا علينا بتغيير الشرط في بدايه الحلقه أي يصبح **while i<5 do**  
قم بتجريب هذه الملاحظه ، لكي تفهمها بشكل جيد ...

اذا لم تكتب هذه الجمله داخل الـ Block // **i:=i+1** فإن الناتج  
عباره عن قيمه المتغير I (1 اذا كنت قد اسندت للمتغير I قيمه قبل بدء  
حلقه **while** ، و 0 اذا لم تسند (او تعطي لها قيمه )  
سوف ينفذ الى ما لا نهايه من المرات .



## النوع الثالث :: حلقه Repeat ::

// الشكل العام //

**Repeat**

:

**Until condition**

### كيفية عملها:

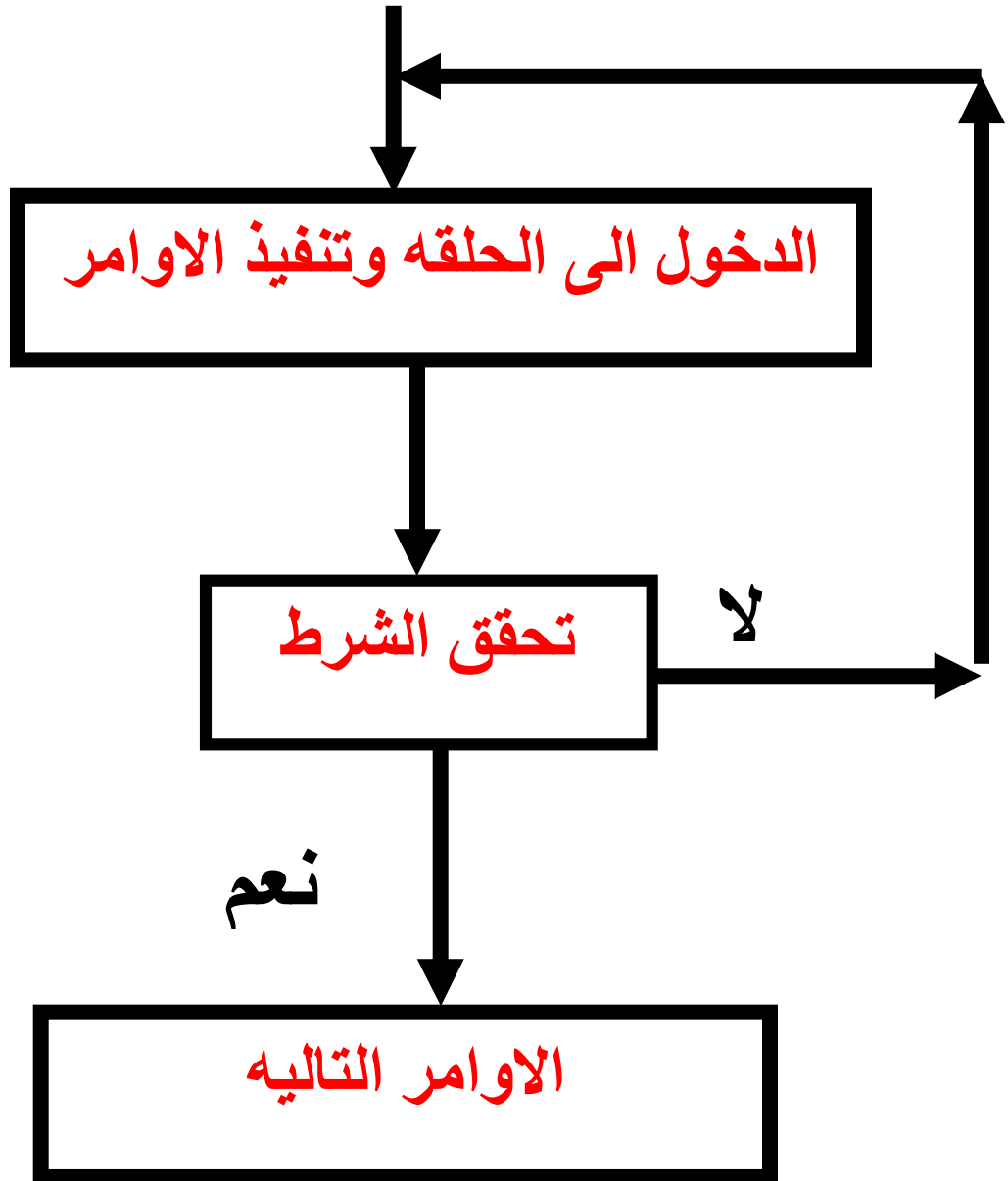
في البدايه يقوم المترجم بدخول الحلقه **repeat** وتنفيذ جميع الاوامر التي بداخله حتى يصل الى جمله **until** فيتوقف ليسأل هل الشرط متحقق ام لا؟؟

اذا نعم فإنه يقوم بالخروج من التكرار !

وإذا لا (الشرط غير متحقق) فإنه يقوم بالدخول الى الحلقه مره اخرى وتكرار الاوامر ... وهكذا ، حتى يتم التحقق من الشرط .

**يجب الملاحظه** ان حلقه **repeat** لا تحتاج الى **begin** ولا **end;**

وتكون كما في الشكل التالي //



مثال // لطباعه الاعداد من 1 الى 5 بحلقه repeat //

**Program** loop3 (input,output);

**Uses** wincrt;

**Var**

I:integer;

**Begin**

I:=1;

**repeat**

Writeln(i);

I:=i+1;

**Until** i>5

**End.**

## شرح البرنامج

في البدايه ( بعد تعريف متغير من النوع **integer** ) تم وضع قيمه ابتدائيه للمتغير I وهي 1 (I:=1) . بدأنا التكرار **repeat** ، سوف يجد المترجم جمله **repeat** معناها أي كرر الجمل الاتيه حتى **Until** يتحقق الشرط وهو I اكبر من 5 فإذا تحقق فاخرج من الحلقة .

في البدايه سيجد هناك جمله طباعه للمتغير I (قيمه تساوي 1 بسبب التعيين الذي قبل جمله التكرار).

بعد ذلك سيجد العبارة I:=i+1 ، ومعناها "اي قم زياده 1 على المتغير I (قيمه تساوي 1) ، وقم بتعيينهما داخل المتغير I " .

اي ان قيمه المتغير I الحاليه تساوي 2 .

الان سيذهب الى جمله **until** ، ويتحقق من الشرط . هل 2 اكبر من 5 ؟ اذا كان الجواب نعم فسوف يقوم بالخروج من الحلقة . اما اذا لا (وهو الصحيح الان) فسوف يرجع مره اخرى للتكرار .

فيطبوع المتغير I (قيمه تساوي 2) ، ويقوم بزياده 1 على المتغير ويصبح 3. والان يتحقق من الشرط .. وهكذا

الى ان تكون قيمه المتغير I هي 5 ، فيطبع الرقم 5 ، ثم يزيد عليها واحد  
(أي ستصبح 6) والان يذهب الى الشرط هل 6 اكبر من 5؟؟  
نعم ، اذا سيخرج من الحلقه .  
أي أن الناتج هو الارقام من 1 الى 5 مطبوعه كل واحد في سطر .

الملاحظات التي تنطبق على **while** ايضا تنطبق على **repeat**، قم بفهمها  
الان ، وقم بكتابه المزيد من البرامج عليهما .

في جمله **while** وجمله **repeat** من الممكن ان يصبح التكرار الى ما  
لانهايه مثلا/

```
While 2<4 do  
Writeln('wajdy');
```

دائما اصغر من 4 ! لذلك سوف يكون التكرار الى ما لا نهايه .

```
Repeat  
Writeln('wajdy');  
Until 3>5
```

كرر حتى تصبح 3 اكبر من 5 !! لذلك سوف يكون تكرار جمله الطباعه  
حتى تصبح 3 اكبر من 5 (بالعربي الى ما لانهايه).

## الحلقات المتدخلة :: Nested Loops

في اغلب الاحيان نحتاج الى مثل هذه الانواع من الحلقات ، أي حلقة For بداخل حلقة for اخرى ، while بداخل while ، ايضا من الممكن مثلا repeat بداخلها حلقة for ، وهكذا .

كل تلك الامور تكون على حسب البرنامج المطلوب ، على مدى تفكير المبرمج ، بمعنى ان من الممكن ان نجد لبرنامج واحد اكثر من 3 طرق لحله لذلك لا تحاول حفظ أي برنامج وانما حاول تفهم المفهوم العام ، ومن ثم ستستطيع كتابه البرامج بطريقتك الخاصه .

مثال // حلقة داخل حلقة ، سوف تكون بالشكل التالي ::

**Repeat ....**

:

**For ..... do**

**Begin**

:

**End;**

:

**Until .....**

**للخروج من الحلقة ::**

نكتب هذا الامر :: **exit**

**If  $i > 3$  then exit;**

**مثال:**

**Begin**

**For  $i := 1$  to 10 do**

**Begin**

**Writeln(i);**

**If  $i = 5$  then exit ;**

**End;**

**Writeln('Done');**

**End.**

أي اذا وصلت قيمه المتغير I 5 فقم بالخروج من هذه الحلقة !!

في المثال السابق تكون **المخرجات هي :**

1

2

3

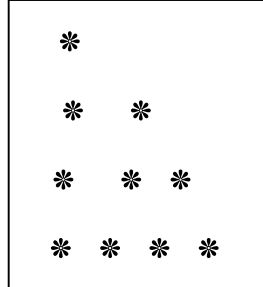
4

5

Done

/// مثال على الحلقات المتداخلة

لطباعة الشكل الآتي :



**Program** loop4 (input,output);

**Uses** wincrt;

**Var**

I,j:integer;

**Begin**

**For** i:= 1 to 4 **do**

**Begin**

**For** j:= 1 to I **do**

Write(' \* ');

Writeln;

**End;**

**End.**

**شرح البرنامج::**

في البدايه قمنا بعمل حلقتين (واحد بداخل اخرى) ، الفكرة هي ان يطبع اربع اسطر (وهي الحلقة الاولى) السطر الاول يطبع فيه نجمة ثم ينتقل المؤشر للأسفل ، والسطر الثاني به نجمتان وينتقل الى السطر الثالث (وهي الحلقة الثانية) و السطر 3 به ثلاثة نجوم ، السطر 4 به اربع نجوم .

## دعنا نبدأ خطوه بخطوه !!

اولا عندما يأخذ المتغير I القيمه الاولى 1 يقوم بتنفيذ ما بين الـ Block .  
الـ block يوجد به حلقه اخرى (داخليه) ، اي ستنفذ الحلقه الداخليه كامله  
عندما تأخذ الحلقه الخارجيه القيمه 1 . وعندما تأخذ الحلقه الخارجيه القيمه 2  
ستنفذ الحلقه الداخليه كامله مره اخرى . وهكذا كل ما تاخذ الحلقه الخارجيه  
قيمه ما فإن الحلقه الداخليه تنفذ كامله .

الان اخذ متغير الحلقه الاولى (الخارجيه) القيمه 1 (اي I يساوي 1 الان).

الحلقه الثانيه من 1 الى I (أي الى 1) أي ستكرر مره واحده الجمله التاليه  
وهي طباعه نجمه ، الان تنفذ عبارته writeln .  
وبهذا انتهى تنفيذ الحلقه عندما تكون قيمه المتغير I تساوي 1

الان عندما تكون قيمه متغير الحلقه الخارجيه I تساوي 2 ، سيقوم بتنفيذ ما  
بين الـ block

الحلقه الداخليه من 1 الى I (أي الى 2) أي ستكرر الحلقه مرتان ،  
المره الاولى سوف تنطبع \* ، والمره الثانيه \* (أي نجمتان في نفس السطر)  
ثم تنفذ عبارته writeln وينتقل المؤشر للسطر الثالث ، وهكذا !!



اليوم السابع

الاجراءات والدوال

**procedure &Function**

كلنا نعلم ان لغه باسكال لغه تركيبيه او هيكلية structured language هذا يعني ان المشاكل تقسم الى خطوات ومشاكل صغيره ، وبعد ذلك تكتب بشكل مرتب ، هذه الترتيبات والتقسيمات تقودنا الى الاجراءات والاقترانات .

بشكل عام الاجراءات والاقترانات هي عباره عن برامج منفصله لها Block خاص بها أي يحتوي على begin و end; .

وهي مفيده عندما يكون هناك قسم من البرنامج يجب ان يقوم باعمال معينه في اجزاء متفرقه في البرنامج !!

**دعنا نلقي نظره على الشكل العام //**

**Procedure procename ;**

**Begin**

:

:

**End;**

**Begin**

:

**End.**

كما شاهدنا ، الاجراءات تعرف قبل بدايه البرنامج الرئيسي (الاقترانات سوف تناولها لاحقا) ايضا مكان procename يتم كتابه اسم الاجراء الذي تريده !!

## فكره الاجراءات ::

في البدايه نقوم بكتابه الاجراء الذي نريد ، ومن ثم في البرنامج الرئيسي  
نقوم باستدعاء الاجراء عن طريق كتابه اسمه فقط !!

## // مثال للاجراءات

**Program** example (input, output);

**Uses** wincrt;

**Procedure** pause;

**Begin**

Writeln ('press any key to continuo ');

Readln;

**End;**

**Begin**

Writeln ('my name is wajdy essam ');

Pause;

Writeln ('I like Pascal Language ');

Pause;

Writeln ('good luck !! ');

Pause;

**End.**

## شرح البرنامج //

في البدايه قمنا بكتابه اجراء يقوم بطباعه جمله في الشاشة ، ويقوم بانتظار المستخدم ان يدخل أي مفتاح وذلك عن طريق الجملة Readln.

ومن ثم بدأنا البرنامج ، طبعا الجملة الاولى على الشاشة ، وتم استدعاء الاجراء pause .

الان سيقوم بتنفيذ الاجراء (أي طباعه الجملة وانتظار ادخال أي مفتاح من المستخدم) . وهكذا في جميع الجمل الباقية ، كل مافي الامر انه يتم استدعاء الاجراء وتنفيذ ما فيه !!!

بدون كتابه الاجراء سيصبح البرنامج السابق :

**Program** example (input, output);

**Uses** wincrt;

**Begin**

Writeln ('my name is wajdy essam ');

Writeln ('press any key to continuo ');

Readln;

Writeln ('I like Pascal Language ');

Writeln ('press any key to continuo ');

Readln;

Writeln ('good luck !! ');

Writeln ('press any key to continuo ');

Readln;

**End.**

الآن ما هو الأفضل؟ أكيد الأول، أوضح وأسهل للقراءه، أليس كذلك!!، هكذا هي الاجراءات والاقترانات تجعل البرنامج سهل القراءه وسهل التتبع ايضا من السهل اكتشاف الاخطاء وتصليحها.

الاجراءات تحتوي على متغيرات خاصه بها تسمى المتغيرات المحليه local variable وهي معرفه فقط داخل الاجراء (أي غير معرفه في البرنامج الرئيسي).

//مثال

**Procedure** wajdy;

**Var**

a: integer;

**Begin**

a:=10 ;

Writeln (a);

**End;**

**Begin**

wajdy;

a:= 20; {هنا خطأ، لان المتغير معرف فقط داخل الاجراء}

**End.**

تم استدعاء الاجراء wajdy، ولكن المتغير a خاطئ، لانه معرف على انه متغير محلي، أي خاص داخل الاجراء فقط!!

المتغيرات العالمية global variable هي المتغيرات المعرفة داخل البرنامج الرئيسي وداخل الاجراءات والاقتوانات .

**Var**

B: integer;

**Procedure** wajdy;

**Var**

A: integer;

**Begin**

A: = 10;

B: =64; {this is legal}

Writeln (a);

**End;**

**Begin**

Wajdy;

A: = 20; {this is illegal}

B: = 50; {this is legal}

**End.**

كما نرى ان المتغير العالمي يمكن تغييره داخل الاجراء والبرنامج ، اما المتغير المحلي فلا يمكن ذلك الا من خلال الاجراء فقط !!

اما اذا تم اعلان المتغير المحلي والعالمي بنفس الاسم ، فإن العالمي سيتجاوز overridden على المحلي ، كما في المثال الاتي //

**Program ex2 (input, output);**

**Uses** wincrt;

**Var**

A: integer;

**Procedure** wajdy;

**Var**

A: integer;

**Begin**

A: =10;

Writeln (a);

**End;**

**Begin**

A: =50;

Writeln (a);

Wajdy;

Writeln (a);

**End.**

الناتج من تنفيذ البرنامج هو :

50

10

50

**الوسائط Parameters :**

عندما نريد قراءه متغير محلي في البرنامج الرئيسي ، يجب في هذه الحاله ان

نستخدم الوسائط psrsmeters ، وهي متغيرات يزود بها البرنامج الرئيسي البرنامج الفرعي لاداء وظيفته .

## كيفية استخدام الوسائط

اولا عند الاعلان عن الاجراء ، نقوم (بعد كتابه اسم الاجراء) بفتح قوس ونضع فيه المتغيرات ، ثم تقطين ( : ) ثم نضع نوع لهذه المتغيرات . وتسمى هذه المتغيرات "بالمغيرات الشكلية" formal variables .

الان في اثناء البرنامج الرئيسي ، عند استدعاء الاجراء نقوم بفتح قوس ونقوم بكتابه متغيرات متشابهه مع المتغيرات الشكلية (التي قمنا بكتابتها بعد اسم الاجراء مباشره) . وتسمى بال "متغيرات الفعلية" Actual Variables .

كلمه متشابهه تعني "ان المتغيرات الفعلية تكون عددها مساو لعدد المتغيرات الشكلية ، وان المتغيرات الفعلية تكون من نفس نوع المتغيرات الشكلية " وبدون تحقق هذين الشرطين لن يتم تنفيذ البرنامج .

كما ذكرنا المتغيرات الشكلية تكون بعد اسم الاجراء (متبوعه بنوعه) . اما المتغيرات الفعلية فيتم تعريفها بشكل طبيعي (اي في قسم الاعلان عن المتغيرات var) .

مثال بسيط :

**Program** ex ( I, o);

**Uses** wincrt;

**Procedure** w (a, b: integer);



```
Var  
X: integer;  
Begin  
Read (a, b);  
X: =a+b;  
Write(x);  
End;
```

```
Var  
no1, no2: integer;  
    Begin  
        W (no1, no2);  
    End.
```

### شرح البرنامج

تم الاعلان عن متغيرين شكليين a,b من النوع الصحيح وذلك في الاجراء W بعد ذلك تم الاعلان عن المتغيرات الفعلية ، وفي البرنامج الرئيسي تم استدعاء الاجراء W مع الوسائط (المتغيرات الفعلية) .  
لاحظ تشابه كل من نوع وعدد المتغيرات الشكليه والفعلية .

### التمرير passes

هناك نوعين من التمرير ، ولكل منهم استخدامه الخاص ، دعنا نرى بعض الامثله :

```
Procedure foo (a: byte);  
Begin
```

```
Writeln (a);    {15}  
a:=10;  
Writeln (a);    {10}  
End;
```

**Var**

```
x: byte;
```

**Begin**

```
x: =15;
```

```
Writeln(x);    {15}
```

```
Foo(x);
```

```
Writeln(x);    {Still 15}
```

**End.**

المخرجات

15

15

10

15

شرح البرنامج

في البدايه كانت  $x=15$  ثم مرر الاجراء الى  $a$  ، وتم تغير  $a$  . وتم طباعته  
10 ، ثم تم طباعه  $x$  (لاحظ لم تتغير قيمه  $x$  بالرغم من تغير قيمه  $a$  في  
الاجراء ) .

هذا النوع من التمرير يسمى " التمرير بالقيمه " pass by value

## المثال الثاني:

**Procedure** foo (var a: byte); {See the difference?}

**Begin**

Writeln (a); {15}

a: =10;

Writeln (a); {10}

**End;**

**Var**

x : byte;

**Begin**

x: =15;

Writeln(x); {15}

Foo(x);

Writeln(x); {10}

**End.**

## المخرجات

15

15

10

10

## الشرح

إذا تم اضافته جمله var امام الوسائط parameters ، فإن القيمة x سوف تتغير اذا تغيرت قيمه a .

هذا النوع من التمرير يسمى " التمرير بالاشارة او المرجع " pass By Reference .

يجب ان نستخدم التمرير بالقيمة في حاله انه غير ضروري ان نغير الوسائط التي في الاجراء .  
اما اذا كان نغير الوسائط ضروري في البرنامج ، نستخدم التمرير بالاشارة .

## الاقتارات :functions

تتشابه كثيرا مع الاجراءات ، الا انها تقوم بإرجاع قيمة في اسم الداله (الاقتران) ، اي يحمل القيمة النهائيه . لذلك يعتبر اسم الداله متغير ويجب كتابه نوعها بجانب اسم الداله ، واذا كانت هناك وسائط فإنها تكتب بعد اسم الداله (كما في الاجراءت) ثم ( : ) ثم نوع الوسائط . بعدها يتم كتابه نوع الداله .

والشكل العام لها //

**Function** funcname (parameters): returntype;

**Begin**

:

:

**End;**

**Begin**

:

**End.**

ايضا الاقتوانات دائما تقع قبل بدايه البرنامج الرئيسي (مثلها مثل الاجراءات).

## نداء الداله

يتم نداء الداله بطريقتين :

عن طريق التعيين ، اي يتم في قسم الاعلان عن المتغيرات (العالميه)  
الاعلان عن متغير جديد وليكن X ، بعدها يتم نداء الاجراء بهذا الشكل :  
X: = functionname;

عن طريق جمله طباعه متغير ، حيث كما ذكرنا يعتبر اسم الداله  
متغير ، ويتم النداء بهذا الشكل :

Write (functionname);

مثال:

```
Program ex (input, output);  
Uses wincrt;  
Function w (a, b: integer): integer;  
Begin  
Read (a, b) ;  
W: = a+b;  
End;  
Var  
no1, no2: integer;
```

**Begin**

Write (w (no1, no2));

**End.**

### شرح البرنامج :

تم تعريف الداله W من النوع الصحيح وتم وضع وسيطين فيها ، والداله تقوم بقراءه عددين ومن ثم جمعهما .

الان في البرنامج الرئيسي تم استدعاء الداله ، وذلك عن طريق طباعه الداله (كما ذكرنا الداله هي عبارته عن متغير) .

بالنسبه للمتغيرات الشكلييه والفعليه ، فهي نفسها الموجوده في الاجراءات .

مثال // لعمل برنامج لحساب المضروب بواسطه الاقترانات :

**Program** factorial (input, output);

**Uses** wincrt;

**Function** fact (n: integer): integer;

**Var**

I,f:integer;

**Begin**

F: =1;

**For** i: = 1 **to** n **do**

F: =f\*I;

Fact: =f;

**End;**

```
Var  
X: integer;  
Begin  
  Write ('please enter any number: ');  
  Readln(x);  
  Writeln (x,' ! is ',fact(x) );  
End.
```

### استدعاء الداول والاجراءات

من الممكن استدعاء اجراء من داخل اجراء اخر ، بشرط ان يكون الاجراء الذي استدعى منه ، ان يكون بعد الاجراء الاول .

```
Procedure a;  
Begin  
:  
End;  
Procedure b;  
Begin  
A; {legal}  
End;  
Begin  
:  
End.
```

يجوز Legal ان يقوم الاجراء B باستدعاء الاجراء A ، ولكن ان يقوم  
الاجراء a باستدعاء الاجراء b ، فهذا خاطئ illegal .

من الممكن ان يقوم الاجراء باستدعاء داله ، او ان تقوم الداله باستدعاء  
اجراء ، ايضا الدوال تستطيع استدعاء بعضها البعض ، مع المحافظه على  
الشرط ، وهو ان تكون الداله الذي يريد الاستدعاء منها بعد الداله الذي يريد  
استدعائها . مثلها مثل الاجراءات .

## الاجراءات المتداخلة

مثل الجمل الشرطيه والتكراريه ، الاجراءات ايضا من الممكن ان تصبح  
متداخلة

**Procedure e; {e cannot access a, b, c, and d}**

**Begin**

:

**End;**

**Procedure a;**

**Procedure b;**

**Begin**

c; {illegal}

e; {legal}

**End;**

**Procedure c;**

**Begin**



b; {legal}

e; {legal}

**End;**

**Begin**

:

b; {legal}

c; {legal}

e; {legal}

**End;**

**Procedure d;**

**Begin**

:

b; {illegal}

c; {illegal}

a; {legal}

e; {legal}

**End;**

**Begin**

:

b; {illegal}

c; {illegal}

a; {legal}

d; {legal}

e; {legal}

**End.**

الاجراء c من الممكن ان يستدعي الاجراء b ، لان c يأتي بعد الاجراء b .  
اما الاجراء b فلا يمكن ان يستدعي الاجراء c لان الاجراء c يأتي بعد  
الاجراء b.

الاجراء a الذي يحتوي على اجريين b و c ويحتوي على block ايضا  
يستطيع ان يستدعي الاجريين b و c .

الاجريين b و c ، لا يمكن استدعائهما من الاجراء d ، ولا من البرنامج  
الرئيسي main block .

إذا : الاجراءات المتداخلة ، لا يمكن استدعائها الا من داخلها ، كما في  
الاجريين b و c .

**الاستدعاء الذاتي**

**Procedure a;**

**Begin**

A;

**End;**

**Begin**

A;

**End.**

. RECURSIVE CALLS ، يسمى هذا الاستدعاء بالاستدعاء التكراري ، وهو يشبه حلقه repeat .. Until

دعنا نأخذ مثالا على ذلك :

**Procedure** m;

**Var**

N: integer;

**Begin**

Readln (n);

**If**  $n \leq 100$  **then** m;

**End;**


**Begin**

Writeln ('please enter any number: ');

M;

**End.**

في البرنامج السابق ، اذا كانت قيمه n اصغر من او تساوي 100 فإنه يتم استدعاء الاجراء m (اي استدعاء نفسه) وهكذا . الى ان تصبح قيمه n اكبر من 100 فإنه يتم انهاء الاجراء .

 النقطة المهمه هي انه في هذا النوع يجب ان نضع شرط ما ، والا سيتم استدعاء الاجراء الى عدد غير نهائي من المرات ، لذلك لا تتسى وضع شرط معين اذا لم يتحقق يتم الخروج من الاجراء (مثلا) .

## اليوم الثامن

### المصفوفات

في الايام السابقه ، عندما تريد الاعلان عن متغير صحيح واحد أو اثنين ، من السهل الاعلان عنهم

اما اذا كان مثلاً 20 متغير من النوع الحرفي !! هل ستعلن عنهم واحد واحد .. ام ماذا !

سوف تتعلم اليوم المصفوفات وطرق الاعلان عنها والمصفوفات الثنائيه والثلاثيه وما الى ذلك

ايضا سوف نتعامل مع المصفوفات (السلاسل) الحرفيه **string** .

## ما هي المصفوفة

المصفوفة array هي مجموعة من مواقع تخزين البيانات، والتي يشتمل كل موقع منها على نفس نوع البيانات . وكل موقع يسمى "عنصر" في المصفوفة وكل موقع (عنصر) له قيمة معينة .

## الإعلان عن المصفوفة

يعلن عنها في قسم الاعلان عن المتغيرات ، بهذا الشكل :

### Var

Arrayname: **array** [x..y] **of** type ;

Type: هي احد انواع المتغيرات .

// مثال

### Var

X:**array** [1..10] **of** integer;

Data:**array**[5..25] **of** char;

يمكن اعلان المصفوفات بأي اسم كما تريد ، ايضا من الممكن ان تكون المصفوفات متغير عالمي او محلي .

لنفترض ان لدينا مصفوفة مكونه من 30 عنصر

### Var

X:**array** [1..30] **of** integer;

هذا يعني ان لكل عنصر في المصفوفه قيمه معينه ،لادخال القيم الى المصفوفه يكون ذلك اما عن طريق التعيين ، او جمله الادخال . كما سيأتي بعد قليل .

### لادخال قيمه عن طريق جمله التعيين :

```
X[1]:=10 ;  
X[2]:= 30;  
:  
X[30]:=5;
```

اذا العنصر الاول اصبح لديه قيمه وهي 10 ، والعنصر الثاني له القيمه 30 وهكذا ..

**مثال //**

```
Var  
X:array [1..10] of integer;  
Begin  
X[1]:= 3;  
X[5]:= 2;  
Writeln(x[1]*x[5]);  
Writeln(x[1]);  
End.
```

## شرح البرنامج

بعد تعريف المتغير x من النوع مصفوفه لها 10 عناصر من النوع الصحيح قمنا بتعيين العنصر الاول للقيمه 3 (يجب ان تكون من النوع الصحيح بسبب اننا عرفنا المصفوفه من النوع الصحيح) .  
وتعيين العنصر 5 للقيمه 2 .  
الان قمنا بطباعه قيمه متغير (النوع الثاني من انواع الطباعه) وهي حاصل ضرب قيمه العنصر الاول في قيمه العنصر الخامس (اي  $2 * 3$ ) .  
اخيرا قمنا بطباعه قيمه العنصر الاول في المصفوفه .  
( لاحظ ان قيم المصفوفه لا تتغير ابدأ ، إلا اذا قمنا باعطائها قيمه جديده عن طريق تعيين او ادخال قيمه ) .

## المخرجات

6  
3

### لادخال القيم عن طريق جمله الادخال ::

بما ان هناك ثلاثين 30 عنصر في المصفوفه ، ونريد قرائتهم جميعا ، فإنه يعتبر مضيعه للوقت ان نقرأ عنصر عنصر هكذا :

```
Read(x[1]);  
Read(x[2]);  
:  
Read(x[30]);
```

لذلك وجب استخدام تكرار بحيث نقوم بقراءة جميع القيم في سطرين فقط!!  
دعنا نرى ذلك ..

```
For i:= 1 to 30 do  
Readln(x[i]);
```

أرأيت ، (باعتبار ان المتغير  $i$  من النوع `integer` ) . سوف يقوم هذا التكرار بقراءة جميع القيم  $x_1$  و  $x_2$  و  $x_3$  ... الى  $x_{30}$  .

مثال // لكتابه برنامج يقرأ 10 احرف ، ثم يقوم بطباعتها بالعكس.؟؟

```
Program ex4 (input,output);  
Uses wincrt;  
Var  
I:integer;  
X:array [1..10] of char;  
Begin  
    For i:= 1 to 10 do  
        Read(x[i]);  
  
    For i:=10 downto 1 do  
        Write(x[i]);  
End.
```



## شرح البرنامج //

قمنا (بعد تعريف المتغيرات ، واحد لعمل الحلقة ، والآخر للمصفوفة ) بعمل مصفوفة مكون من 10 عناصر . كل عنصر له قيمة معينة حيث قمنا بكتابه كل عنصر من خلال حلقه For الاولى . وفي الحلقة الثانيه قمنا بطباعه العنصر العاشر ثم التاسع (أي بالعكس) لاحظ جمله **downto** ، وهكذا ..

## متى تكون هناك الحاجه للمصفوفات ؟

لنفترض ان لدينا قائمه من الاسماء لنفترض 10 ، سيكون تعريفها بهذا الشكل //

### Var

```
Name1,nam2,name3,name4,name5,name6,name7,  
name8,name9,name10:string ;
```

اما بالمصفوفات سوف يكون الاعلان كالتالي :

### Var

```
name:array[1..10] of string;
```

## المصفوفه الثنائيه الابعاد :two-dimensional array

يكون الاعلان عنها بهذا الشكل //

### Var

**X:array[1..10 , 1..5] of byte ;**

ويتم ادخال القيم اليها هكذا :

**x[5,2]:= 20;**

اي ان في الصف الخامس والعمود الثاني ، القيمه تساوي 20 .

وهكذا في المصفوفه الثلاثيه الابعاد ، والرابعيه ، الى 100 بعد .  
كلها بنفس المفهوم !!

الذي تحتاجه انه في المصفوفه من البعد الثاني ستحتاج الى حلقتين الاولى  
للبعد الاول والثانيه للثاني ، ايضا المصفوفه ذات الثلاثه ابعاد ستحتاج الى  
ثلاث حلقات متداخله !! (هذا اذا كنت تريد ادخال قيم الى المصفوفه عن  
طريق جمله الادخال read و readln).

مثال // اكتب برنامج لضرب مصفوفتين مربعتين !!؟

**Program** example (input,output);

**Uses** wincrt;

**Var**

I,j,sum:integer;

**X,a,b:array** [1..3,1..3] of integer;

**Begin**

**For** i:= 1 to 3 do

**Begin**

```
For j:=1 to 3 do  
  Read(a[I,j]);  
End;
```

```
For i:= 1 to 3 do  
Begin  
  For j:=1 to 3 do  
    Read(b[I,j]);  
  End;  
  sum:=0;
```

```
  for i:=1 to 3 do  
  begin  
    for j:= 1 to 3 do  
      x[I,j]:=sum + ( (a[I,j]) * (b[I,j]) ) ;  
    end;
```

```
  for i:= 1 to 3 do  
  begin  
    for j:= 1 to 3 do  
      writeln(x[I,j]);  
    end;  
  end.
```

## شرح البرنامج

اولا ماذا تعني مصوفه مربعه ؟؟ اي مصفوفه ذات بعدين .. اي انها تتكون من بعدين (انتبه جيدا )

وفي المصفوفات ذات البعدين "يجب ان يكون هناك حلقتين متداخلتين" ليتم قراءه البعد الاول والبعد الثاني .  
في المثال البعد الاول هو I والثاني هو j .

الان ، دعنا نبدأ بعد الـ **begin** الرئيسي :

```
For i:= 1 to 3 do
```

```
  Begin
```

```
    For j:=1 to 3 do
```

```
      Read(a[I,j]);
```

```
    End;
```

هنا سيتم قراءه المصفوفه a كامله (البعد الاول والثاني) .

```
For i:= 1 to 3 do
```

```
  Begin
```

```
    For j:=1 to 3 do
```

```
      Read(b[I,j]);
```

```
    End;
```

هنا سيتم قراءه المصفوفه b كامله (البعد الاول والثاني) .

sum:=0; عبارہ عن مخزن ، سيتم فيه جمع قيمه ضرب المصفوفه الاول والثانيه ، ويتم تخزينها في مصوفه اخرى ثالثه X .

```
for i:=1 to 3 do
begin
  for j:= 1 to 3 do
    x[I,j]:=sum + ( (a[I,j]) * (b[I,j]) ) ;
  End;
```

الان سيتم ضرب المصفوفه الاولى a في الثانيه b ويتم تخزين الناتج في المصفوفه الثالثه X .

```
for i:= 1 to 3 do
begin
  for j:= 1 to 3 do
    writeln(x[I,j]);
  end;
```

الان سيتم طباعه المصفوفه الثالثه X .

## التعامل مع السلاسل النصيه:

فعليا ، السلسله النصيه هي مصفوفه من الاحرف char ، لنفترض ان المتغير s من النوع string ، العنصر الاول s[1] يساوي الحرف الاول ، والعنصر الثاني s[2] يساوي الحرف الثاني .. وهكذا .

**Program example (input,output);**

**uses** wincrt;

**var**

**s : string;**

**begin**

s:='Hello, dear';

writeln(s);

s[1]:='J'; {Replace the first character with J}

s[5]:='y'; {Replace the fifth character with y}

writeln(s); {Jelly, dear}

writeln('The length of s is ',ord(s[0]) );

**end.**

## شرح البرنامج

بعد تعريف المتغير s من النوع السلسله نصيه string ، قمنا بإسناد هذا المتغير الى الجملة Hello, dear (لاحظ القوس الصغير، وهكذا عند اسناد المتغيرات من النوع string والنوع char ، يجب وضع القوسين الصغيرين).

ثم قمنا بطباعه المتغير s (والذي قيمته Hello, dear ) وذلك بالعباره  
writeln(s) .

الآن قمنا بتعيين قيمه جديده للعنصر الاول في السلسله النصيه وذلك بالعبارة  
s[1]='J' اي قمنا باستبدال الحرف الاول H بالحرف J .  
ايضا قمنا بتعيين قيمه جديده للعنصر الخامس في السلسله النصيه ، وذلك  
بالعبارة s[5]='y' ، اي قمنا باستبدال الحرف الخامس o بالحرف y .

الآن قم بطباعه المتغير s (بعد عمليات الاسناد ) ولذلك بالعبارة writeln(s)

الآن العبارة الاخير ، هي نوع من انواع الطباعة (النوع الثالث بالتحديد) ،  
حيث يقوم بطباعه نص وقيمه لمتغير (مع الفصل بينهم بفاصله).  
النص هو The length of s is .  
قيمه المتغير هي ord(s[0]) . ومعناها "طول السلسله النصيه" .

وهناك داله اخرى في باسكال ، تستطيع ان تستخرج منها الطول بكل يسر ،  
وسوف يتم التحدث عنها بعد قليل .

اما بالنسبه للـ ord ، فله استخدامان وسيتم شرحهم في الدروس القادمه  
بالتفصيل .

عاده ، السلسله النصيه تأخذ 80 حرف كحد اقصى ، اذا كنت تريد الاعلان  
عن سلسله نصيه لكن بحدود 10 احرف مثلا ، باسكال تقدم وسيله لتحديد عدد  
الاحرف . لان الاعلان عنها بالطريقه العاديه سوف يكون تبذير في المساحه

**Var**

**S:string[10];**

## الوامر التي تتعامل مع السلاسل النصيه

### 1/ الامر length

يستخدم لاستخراج طول السلسله (كما في المثال الاول ،  $\text{ord}(s[0])$  ) ولهما نفس النتيجة .

الصيغه العامه :  $\text{length}(s)$

(باعتبار s متغير من النوع string ، ايضا في جميع الوامر التاليه ) .

```
S:= ('wajdy essam');
```

```
N:= length(s);
```

```
Writeln(n);
```

المخرجات هي : 11

### 2/ الامر Copy

يقوم بنسخ الحروف من مكان محدد from ، وكم حرف يجب أن يأخذ howmuch .

الصيغه العامه :  $\text{copy}(s,\text{from},\text{howmuch})$

```
S:= ('wajdy essam');
```

```
N:= copy(s,7,3);
```

```
Writeln(n);
```

معنى السطر الثاني ، قم بالبدء بعملية النسخ من الحرف 7 ، وانسخ 3 فقط .

المخرجات هي : ess



إذا كانت البدايه في عمليه النسخ from اكبر من السلسله ، فإن الناتج جمله فارغه .

```
S:= '(wajdy essam)';  
N:= copy(s,15,3);  
Writeln(n);
```

لا توجد مخرجات (جمله فارغه).

إذا كانت عدد الحروف التي نريد نسخها howmuch اكبر من السلسله فإنها تكتب الباقي من السلسله .

```
S:= ('wajdy essam)';  
N:= copy(s,7,10);  
Writeln(n);
```

المخرجات هي : **ess**

### 3/ الامر pos

يقوم باستخراج موقع السلسله الصغيره ، من السلسله الكبيره s .  
الصيغه العامه: pos(substr,s)  
Substr هي كلمه البحث  
S السلسله التي يتم البحث بها.

```
S:= pos('are','how are you');  
Writeln(s);
```

المخرجات هي : **5**

حيث تكون الكلمه are في الموقع الخامس في السلسله الكبيره .

## 4/ الامر Insret

يقوم هذا الامر باضافه سلسله نصيه الى اخرى .  
الصيغه العامه : insert(source,target,index)

Source : السلسله الاصل التي نريد ادخالها الى سلسله اخرى  
Target : السلسله المراد الادخال اليها  
Index : الموقع المراد الادخال منه .

```
S1:='not';  
S2:=' I do love you';  
Insert(s1,s2,6);  
Writeln(s2);
```

المخرجات هي : **I do not love you**

اذا كان الناتج اكثر من 255 حرف ، فانه يتم حذف الباقي من  
الـ255 حرف .

## 4/ الامر delete

يقوم هذا الامر بحذف الحروف من السلسله  
الصيغه العامه : delete(s,from,howmuch)

```
S:=('wajdy essam');  
Delete(s,1,6);  
Writeln(s);
```

المخرجات هي : **essam**

## اليوم التاسع

الجميل المعرفة (type)

والمجموعات set

في هذا الدرس سوف تتعلم العديد من مميزات لغة باسكال وهي الجميل  
المعرفة من قبل المبرمج.

ويأتي معها السجلات ، حيث ان السجلات تنطوي داخل انواع الـ  
. type

اما بالنسبة للمجموعات فهو درس قصير وسهل جدا (ايضا هو ينطوي  
تحت انواع الـ type ) ، لذلك تمت اضافته الى هذا اليوم ..

Have Anice Day

## الجمل المعرفة Type

هي عباره عن قيم يتم تخزينها تحت اسم (متغير) . فمثلا الاعداد الصحيحه هي الاعداد من -32768 الى +32767 يتم تخزينها تحت اسم integer . وبامكانك انت تعرف مجموعه وتضعها تحت اسم (متغير) . ويمكن استخدام هذا المتغير في برنامجك بشكل طبيعي ، باستثناء انك لا تستطيع استخدام جمل الطباعه write و writeln وايضا لا تستطيع استخدام جمل القراءه read و readln .

مثلا // تستطيع تعريف مجموعه من الاسامي تحت متغير name :

### Type

Name= (wajdy, essam, Ahmed);

الان اصبح لدينا متغير من النوع name (مثله مثل integer و real و char وباقي المتغيرات) يتم الاعلان عنه في قسم المتغيرات

Var

N: name;

أرأيت ، تماما مثل بقية المتغيرات !!

// مثال

### Type

Car = (bmw , Mazda ,Honda ,ford);

Color = (red ,green ,blue);

Str20 = string[20];

## Var

```
Mycar:car;  
Mycolor:color;  
Name:str20;
```

رائع ، هذه العبارات تسمى **enumerated types** اي الجمل المعرفه مسبقا من قبل المبرمج .

ف نجد ان المتغير mycar من النوع car ، والمتغير mycolor من النوع color .

ايضا المتغير mycolor ، لا يمكن ان يخرج خارج اطار قيمه ، بمعنى ان المتغير ( **عند اسناده مثلا** ) يجب ان تكون قيمه الاسناد هي احد القيم التي يحملها المتغير الجديد color وهي red و green و blue .

فكما ان المتغير من النوع الصحيح integer لا يقبل الاعداد الكسريه 2.6 .  
فالمتغير mycar لا يقبل الا احد عناصره وهي ( , Honda, mazda, bmw , ford ) ، وهكذا .

مثال (تابع البرنامج السابق) :

## Begin

```
Mycar := 1; {illegal}
```

```
Mycar := ford; {legal}
```

```
If mycar = mazda then writeln('I love mazda'); {legal}
```

## End.

- {illegal} معناها (غير شرعي) اي جمله غير صحيحه (مرفوضه) .
- {legal} معناه (شرعي) اي جمله صحيحه (مقبوله) .

// هناك بعض الاوامر التي تتعامل مع الـ Type  
**1. الامر ord :**

له وظيفتان الاولى هي تحديد موقع الجمل المعرفه (في حاله الـ Type)  
 الثانيه هي اعطاء قيمه الحرف بصيغه اسكي ASCII  
 مثال على الوظيفه الاولى (في البرنامج السابق) //

Ord(bmw); {=0}

Ord(mazda); {=1}

مثال على الوظيفه الثانيه //

N:= Ord(a); {n=65}

**2. الامر chr :**

يقوم باعطاء الرقم للحرف المدخل ، طبقا لمعايير ASCII اي عكس الداله

ord

مثال/

Chr(65); {=a}

مع ملاحظه ان قيمه الحرف الـ capital تختلف عن الـ small.

**3. الامر succ والامر pred //**

الاول يقوم باعطاء القيمه او الجمله التاليه (في الجمله المعرفه)  
 الثاني يقوم باعطاء القيمه أو الجمله السابقه (في الجمله المعرفه)

//مثال

**Type**

Color= (red ,green ,blue);

**Var**

Mc : color;

**Begin**

Succ(red); {green}

Pred(blue); {green}

Ord(blue); {2}

**End.**

// ايضا انظر لهذه الخاصيه

**Type**

Car = (bmw=1,mazda ,Honda ,ford=7,Volvo);

**Begin**

Ord (bmw); {=1}

Ord (mazda); {=2}

Ord (Honda); {=3}

Ord (ford); {=7}

Ord (Volvo); {=8}

**End.**

اظن انها وضحت ، تماما !!

## السجلات record

كما في المصفوفات ، السجلات هي عبارة عن عدد معين من المتغيرات تحت اسم واحد ، على كل حال المصفوفات تستطيع ان تحمل نوع واحد من البيانات ، لكن السجلات من الممكن ان تحمل جميع انواع البيانات (المتغيرات) بما فيها المصفوفات !!

فعليا السجلات هو نوع من انواع البيانات المعرفه ، وتتطوي تحت قسم الـ type ، وفي اغلب الاحيان الاعلان عن الـ type يكون قبل الاعلان عن المتغيرات .

نظرة حول برنامج شبه متكامل بباسكال :

**Program** progname (input,output);

**Uses** unit1 , unit2 ....

**Type**

:

**Var**

:

**Procedure & function**

:

**Begin** {main}

:

**End.**



مثال على الاعلان عن السجلات //

**Program** example (I,o);

**Uses** wincrt;

**Type**

Data = **record**

    Name : **string**;

    Age : integer;

    Tel : integer ;

**end**;

**Var**

Da: data ;

السجل لن يستخدم بعد الاعلان عنه ، الا اذا كان هناك متغير da يحمل اسم السجل ، هذه هي اهم نقط في الاعلان عن السجلات .

نأتي الى البرنامج الرئيسي ، وكيفيه ادخال القيم اليها ، عن طريق جمل التعيين او جمله الادخال read و readln .

👉 جمله التعيين (تابع البرنامج السابق) ::

**Begin**

    Da.name := 'wajdy' ;

    Da.age := 20;

    Da.tel := 0911119415 ;

**End.**

## شرح البرنامج :

في الاعلى قمنا باسناد قيم السجل ( الحقول fields ) الى اي قيمه تريد ،  
فمثلا في الحقل الاول name قمنا باسناده الى الجمله wajdy ، والحقل الثاني  
age قمنا باسناده الى الرقم 20 وهكذا .

لاحظ انه يجب ان يكتب المتغير الذي يحمل السجل da ثم نقطه ، ثم اسم  
الحقل المراد تعيينه الى قيمه .

(وكما ذكر سابقا ، عند اسناد السلاسل النصيه والاحرف يجب ان تضع القيم  
بين علامه ' here ' ).

✚ جمله الادخال (تابع البرنامج السابق) ::

### Begin

```
Read(da.name);
```

```
Read(da.age);
```

```
Read(da.tel);
```

### End.

## شرح البرنامج

يتم كتابه جمله القراءه ثم المتغير (الذي يحمل السجل) ثم نقطه ثم اسم الحقل  
المراد . وهكذا ..

هناك طريقه اخرى لاسناد القيم و قراءه (ادخال قيم) السجلات ، وهي افضل  
من الاولى (من ناحيه سهوله فقط لا اكثر) وهي باستخدام **with ... do**

// يكون الاسناد بها بهذا الشكل //

**Begin**

**With da do**

**Begin**

Name := 'wajdy' ;

Age := 20;

Tel := 0911119415

**End;**

**End.**

// يكون ادخال القيم بهذا الشكل //

**Begin**

**With da do**

**Begin**

Read(name);

Read(age);

Read(tel);

**End;**

**End.**

اي يتم في البدايه كتابه الكلمه **with** ، ثم نضع المتغير الذي يحمل السجل ،  
ثم نضع كلمه **do** . بعدها نقوم بفتح block ونضع بداخله الجمل المراد  
ادخالها . اما عمليات اسناد ، او جمل ادخال قيم (كما في المثالين السابقين) .

// اما بالنسبه للطباعه

Writeln (da.name);      {بالطريقه الاولى}

**With da do**

**Begin**

Writeln(name);      {بالطريقه الثانيه}

**End;**

ايضا من الممكن ان تستخدم مصفوفه من السجلات ، نرجع الى المثال الاول  
بعد ان عرفنا السجل data ، الان في قسم المتغيرات سوف يكون الاعلان:

Da : array [1..10] of data ;

// وباسناد قسم المصفوفه

Da[1].name := 'wajdy';

// with ... do بالطريقه الثانيه

**With da[1] do**

**Begin**

Name := 'wajdy';

**End;**

ومن الممكن استخدام حلقه (تكرار) لقراءه جميع قيم السجل (بعد وضعه في  
مصفوفه ) !!

مثال // اكتب برنامج يقرأ معلومات 3 طلاب ومن ثم يطبع هذه المعلومات على الشاشة؟؟ علماً بأن معلومات الطالب هي (الاسم ، العمر ، الهاتف)؟؟

**Program** database (input,output);

**Uses** wincrt;

**Type**

**Data = record**

**Name:string;**

**Age,tel:integer;**

**End;**

**Var**

**Da:array [1..3] of data ;**

**I:integer;**

**Begin**

**Writeln('please fill the information : ');**

**For i:= 1 to 3 do**

**Begin**

**Writeln('NOW .. student no ',i);**

**With da[i] do**

**Begin**

**Write('Name: '); Readln(name);**

**Write('age: '); Readln(age);**

**Write('tel: '); Readln(tel);**

**End;**

**End; {for}**

```

        Clrcsr;
    For i:= 1 to 3 do
        Begin
            With da[i] do
                Begin
                    Writeln(' student no ',i);
                    Write ('name: '); Writeln(name);
                    Write ('age: '); Writeln(age);
                    Write ('tel: '); Writeln(tel);
                End;
            Writeln;
        End; {for}
    End.

```

### شرح البرنامج

في البدايه تم تعريف متغير معرف مسبقا (اي انه هو احد انواع الـ type) من نوع record (اي سجل) باسم data ، وتم وضع ثلاث حقول ، وهي المطلوبه في المثال ، الاسم name ، العمر age ، الهاتف tel .

الان اصبح لدينا متغير من النوع data .

في قسم الاعلان عن المتغيرات ، تم الاعلان على المتغير da والذي هو عباره عن مصفوفه مكونه من ثلاثه عناصر (data) . ومتغير I من النوع الصحيح (يستخدم في التكرار) .

بعد بدايه begin البرنامج الرئيسي نشاهد :

```
Writeln('please fill the information : ');
  For i:= 1 to 3 do
    Begin
      Writeln('NOW .. student no ',i);
      With da[i] do
        Begin
          Write('Name: '); Readln(name);
          Write('age: ');   Readln(age);
          Write('tel: ');   Readln(tel);
        End;
      End;
    End; {for}
```

السطر الاول ، هو عبارته عن جملة طباعه نص ، سوف ينطبع على الشاشة فور تنفيذ البرنامج .

السطر الثاني هو عبارته عن حلقه سوف تكرر ثلاث مرات الفائده منها ؟ هي انه سيتم من خلالها ادخل بيانات (الاسم ، العمر ، الهاتف) ثلاث طلاب .

تحتوي على على begin و end ، اي انها حلقه مركبه واي اوامر في هذه الحلقه سوف تكرر ثلاثه مرات .

الان سوف نرى جملة طباعه نص + قيمه متغير I ، ثم بدأ التعامل مع السجل بالجملة **With da[i] do** ، اي مع الطالب (الاول ، عندما تكون قيمه المتغير I تساوي 1 ، والثاني عندما تكون قيمه I تساوي 2، وهكذا) .

وكما ذكرنا سابقا عند استخدام جملة `Do .. with` يجب ان توضع داخل `block` منفصل ، وتم وضع داخل هذا الـ `block` جملة طباعه على الشاشة وجملة ادخال بيانات .

سوف يتكرر هذا الـ `block` ثلاث مرات (نظرا لـ `for` حلقة) . وهكذا تم ادخال بيانات ثلاث طلاب .

الجملة `clrscr` معناها `clear screen` اي تنظيف الشاشة ، اي سيتم عند كتابه هذا الامر مسح كافة البيانات من الشاشة .

الحلقة الثانيه هي لطباعه بيانات الطلاب الثلاثه ، وهي شبيهه بالحلقة السابقه (الاولى) لذلك لن نعيد تكرار نفس الكلام .

## المجموعات Sets

المجموعات من احد المزايا الخاصه للغه باسكال ، والتي نادرا ما نجدها في لغات البرمجه الاخرى .

اغلب المبرمجين يتجاهلوا هذه الخاصيه ويقوموا بدلا عن ذلك باستخدام المصفوفات ، لما يوجد من تشابه بينهما .

على العموم استخدام المجموعات في برامجنا يعتبر شيء جيد ، جدا !!

انا شخصيا نادرا ما استخدمها ايضا ، لكنها مفيده في بعض الاحيان ، والعديد من الكتب تم تجاهل هذه الخاصيه وعدم الاشاده لها ، لكن ان شاء الله سأحاول أن اغطيها بقدر الامكان .



المجموعه Set في البرمجه هي تقريبا تشابه المجموعه في الرياضيات ،  
دعنا نشاهد المجموعه في الرياضيات //

$S = \{1, 5, 8, 9\}$

وهكذا هي المجموعه في البرمجه (لها بدايه ونهايه ، وجميع عناصرها معلومه) .

دعنا نشاهد كيفيه الاعلان عنها //

**Type**

Days = (Sut,sun,mon,tues,wed,thurs,fri);

**Var**

Allday : **set of days** ;

المتغير allday ، عناصره جميع الايام من السبت الى الجمعه .

//مثال

**Program** ex (input, output);

**Uses** wincrt;

**Type**

Days = (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday);

**Var**

allday : **set of days**;

workday : **set of** Monday .. Friday;

thisday : days;

## Begin

thisday:= Monday;

**If** thisday **in** workday **then**

writeln('This day I'm in work')

**Else**

Writeln ('This day I'm on holiday');

## End.

- . thisday:= Saturday; الى thisday:= Monday; الان قم بتغيير وشاهد الفرق ، اظن انها واضحه جدا !!

لنفترض اننا نريد ان نحذف احد هذه الايام لسبب ما ، يكون ذلك هكذا

Exclude(myday,Friday);

الان تم استبعاد يوم الجمعه من ايام الاسبوع.

ولارجاع او اضافته يوم ، يكون ذلك هكذا

Include(myday,Friday);

**ملاحظه مهمه // الامرين exclude و include لا يعملان في النسخه تيربو باسكال .**

## اليوم العاشر

### عمل وحداتك الخاصه

في الكثير من الاحيان ، نحتاج الى العديد من الاجراءات والدوال وبعض الاوامر ، ان تتكرر في كل برنامج . لذلك عند كتابه هذه الاجراءات والدوال سوف تهدر الكثير من الوقت ، وربما تخطئ في كتابته.

لذلك من الممكن ان تضع جميع اوامرك التي تريدها تحت وحده واحده وتضعها في كل برنامج .

في هذا اليوم ستتعلم كيفيه صناعه الوحدات ، واستخدمها في برنامجك .

## عمل الوحدات الخاصه

الوحده هي عباره عن مجموعه من الاوامر التي تكون موجوده في اسم واحد الا وهو "وحده unit" ، وبدون الاعلان عن اسم هذه الوحده لا نستطيع تشغيل البرنامج .

مثلا الامر write و writeln هما امران موجودان داخل الوحده wincrt ، واذا حاولت كتابه هذه الاوامر بدون الاعلان عن الوحده wincrt فإن البرنامج لن يعمل .

حسنا ، دعنا نرى الشكل العام لكيفيه كتابه وحده خاصه بنا

{This code must be saved with name wajdy.PAS}

**Unit** wajdy;

**Interface**

**Uses....**

**Var**

... {Visible global variables}

{Procedures & functions definitions}

**Procedure** myproc;

**Function** myfunc: word;

:

```
:  
Implementation  
Var  
... {Invisible global variables}  
  
Procedure myproc;  
Begin  
: {the routines}  
:  
End;  
  
Function myfunc: word;  
Begin  
: {the routines}  
:  
End;  
  
:  
: {other procedure's / function's routines}  
:  
  
Begin  
: {Initialization code}  
:  
End.
```

الكلمه المحجوزه **intercafe** يتم الاعلان عن جميع المتغيرات والوحدات واسامي الاجراءات والدوال ، وسوف تكون متاحه لمستخدم وحدتك في اي برنامج .

بمعنى ان اي برنامج يتم وضع الوحده الخاصه بك ، سوف يكون من الجائز تغير المتغيرات في البرنامج .

الكلمه المحجوزه **implementation** تحتوي على محتويات الاجراءات والدوال ، المتغيرات في هذا الجزء لن يتم تغيرها .

مثال بسيط

```
{This unit must be stored in MYUNIT.PAS}
```

```
Unit myunit;
```

```
Interface
```

```
Uses CRT;
```

```
Var
```

```
  X : byte;
```

```
  Y, z: integer;
```

```
Procedure a;
```

```
Procedure b;
```

```
Function c: byte;
```

```
Implementation
```

```
Var
```

P, q, r: shortint;

**Procedure a;**

**Begin**

:

:

**End;**

**Procedure d;**

**Begin**

:

:

**End;**

**Procedure b;**

**Begin**

:

:

**End;**

**Function c: byte;**

**Begin**

:

:

**End;**

**Procedure e;**

**Begin**

:  
:

**End;**

**Begin**

: {Initialization code}

:

**End.**

الآن لعمل برنامج يستخدم الوحدة السابقه

**Uses myunit;**

**Var**

N: byte;

**Begin**

A; {legal}

B; {legal}

N: =c; {legal}

X: =1; {legal}

Y: =-1; {legal}



Z: =14; {legal}

D; {illegal, because it is invisible}

E; {illegal, because it is invisible}

P: =-1; {illegal, because it is invisible}

**End.**

كما ذكرنا في بدايه الدرس ، ان الحاجه لها عندما نريد استخدام العديد من الاوامر او الاجراءات في العديد من البرامج ، نقوم بكتابه وحده تضم هذه الاوامر جميعا . ثم في البرنامج الرئيسي نقوم باضافه هذه الوحده .

اليوم الحادي عشر

الملفات النصيه والثنائيه

## **Text & Binary Flies**

## الملفات النصيه

الملفات النصيه هي عباره عن اي ملف محتوياته عباره عن نصوص text وغالبا ما تكون امتداده txt ، في هذا الدرس سوف نناقش كيف يمكن الكتابه في الملفات النصيه والقراءه ايضا.

في حاله التعامل مع الملفات النصيه ، سنناقش على اساس ان الملف النصي الذي سننشئه (او الموجود) سوف يكون في مجلد البرنامج ، اي موجود في القرص C داخل المجلد TPW .

توجد خطوات معينه في الملفات النصيه ، ويجب الترتيب فيها .

1. اول خطوه هي تعريف متغير ما (وليكن f) من النوع النصي text

**Var**

F: text;

2. ثاني خطوه تكون في تعيين assign المتغير f في اي ملف نصي (موجود مسبقا ، او ملف نصي جديد) ويكون بعد بدايه Begin البرنامج .

في حاله ملف نصي موجود مسبقا (كما قلنا في البدايه داخل

مجلد البرنامج اي في المسار C:/TPW) سوف يكون الـ

assign بهذا الشكل، باعتبار ان الملف الموجود باسم wajdy.txt

**Begin**

Assign (f,'wajdy.txt');

اما في حاله نريد ان ننشئ ملف نصي جديد (اي مستخدم البرنامج هو الذي يكتب الاسم) يجب ان نضع متغير ما وليكن s لكي يحفظ فيه اسم الملف الذي يريده المستخدم

**Var**

F: text;

S: string;

**Begin**

Read(s);

Assign (f, s);

لاحظ المتغير s من النوع string اي سلسله نصيه (لان اسم الملف سوف يكون عبارته عن مجموعه من الحروف والارقام).

3. الخطوه الثالثه هي فتح الملف النصي الموجود مسبقا

Reset (f);

اما في حاله انشاء ملف جديد

Rewrite (f);

4. الخطوه الرابعه هو قراءه البيانات (في الملف القديم) ويتم ذلك بـ

Readln (f, x);

Writeln(x);

المتغير X يجب ان يكون من النوع string ، ومعنى هذه الجملة ان يقوم المترجم بقراءة السطر في الملف وتخزينه في المتغير X، ومن ثم طباعه المتغير X على الشاشة .

اما في حالة ادخال البيانات لملف جديد

Readln(x);

Writeln (f, x);

اي قم بقراءة المتغير X (من النوع string) ومن ثم قم بطاعته في الملف F

5. الخطوة الخامسة هي لمعرفة هل انتهى الملف النصي ، اي هل انتهى المترجم من قراءة الملف النصي ووصل الى نهايه الملف

Eof (f);

و هي اختصار لـ end of file

If eof (f) then writeln ('The End');

6. الخطوة السادسة هي للاغلاق الملف النصي

Close (f);

**مثال على قراءة ملف مسبقا في مجلد البرنامج باسم romansy.txt**

**Program** ex1 (input, output);

**Uses** wincrt;

**Var**

F: text;

S: string;

**Begin**

Assign (f, 'romansy.txt');

Reset (f);

**Repeat**

Readln (f, s);

Writeln(s);

**Until** eof (f);

Close (f);

**End.**

## شرح البرنامج //

في البدايه قمنا بتعريف متغيرين ، الاول f من النوع النصي text والثاني متغير s من النوع string .  
الان بدأنا البرنامج begin ، قمنا بتعيين الملف romansy بالمتغير f .

ومن ثم قمنا بفتح الملف بالعباره reset(f) ، وبدأ بعدها حلقه repaet معناها ان يقوم بقراءه السطر الاول في الملف النصي ويقوم بتخزينه في المتغير s ويقوم بطباعه المتغير s ، ثم يصل الى الجملة Until eof(f); فيقوم بالتحقق من الشرط (اي هل وصل الملف الى نهايته) ام لا .  
اذا نعم (اي وصل) يقوم بالخروج من الحلقه ، اذا لا فيقوم بتكرار هذه الحلقه حتى تتحقق الحلقه (اي يصل الى نهايه الملف) .

الفائده من هذه الحلقة // هي قراءه الملف النصي كاملا من اوله الى اخره وطبعاً في حاله الملفات النصيه ، لا يمكن استخدام الحلقه for لان عدد السطور غير معلوم ، فيجب ان نستخدم جمله تكرر تستخدم الشرط ، وهما while و repeat .

في المثال السابق ، لو ادركنا ان يكون اسم الملف النصي هو الذي يدخله المستخدم ، يكون شكله كالآتي /

**Begin**

Read (n);

Assign (f, n);

:

**End.**

المتغير N هو متغير من النوع string .

مثال على ادخل بيانات في ملف جديد (اي يتم انشاء ملف باي اسم ومن ثم تتم كتابه بعض البيانات عليه).

**Program ex2** (input, output);

**Uses** wincrt;

**Var**

F: text;

S, n: **string**;

Ch: char;

**Begin**

```
Read (n);  
Assign (f, n);  
Rewrite (f);  
Repeat  
Readln(s);  
Writeln (f, s);  
Ch: = readkey;  
Until ch=#27  
Close (f);
```

**End.**

### شرح البرنامج //

بعد تعريف المتغيرات ، F للملف النصي ، n لاسم الملف المدخل من قبل المستخدم ، s لطباعه البيانات المدخلة من المستخدم الى الملف النصي المتغير ch هو عبارته عن متغير حرفي char وسوف اتكلم عنه بعد قليل .

بدأ البرنامج ، قمنا بقراءة المتغير n وعيناه assign الى الملف النصي ،  
وقمنا بإنشاء rewrite .

الآن بدأت حلقة التكرار ، اقرأ المتغير s (**Readln(s);**) ومن ثم اطبعه على  
الملف النصي (**Writeln(f,s);**)  
و (**Ch:= readkey;**) ماذا يقصد بها .....؟



اولا يجب ان نعرف readkey هو عبارته عن داله تستخدم لقراءه (الحرف او الرقم او الرمز) المدخل من المستخدم .  
وقمنا بتعيينه (اسناده) الى المتغير ch .  
وسوف تتكرر هذه الحلقة (اي قراءه المتغير s وطباعته في الملف النصي) حتى تتحقق الحلقة ch=#27 اي يصبح المتغير Ch يساوي القيمه #27 ومعناه المفتاح Esc الموجود في اعلى لوحة المفاتيح من الجهه اليمنى .  
معنى ذلك سوف تكرر الحلقة حتى يقوم المستخدم بالضغط على مفتاح الهروب Esc .

## جمله //append

اذا قمنا بانشاء ملف جديد بنفس الاسم لملف قديم اي منشأ مسبقا ، في هذه الحاله سوف يتم مسح كافه البيانات من الملف القديم .  
اذا جمله append تستخدم لاضافه بيانات جديده الى ملف قديم ، ويتم كتابتها بدل جمله rewrite لاحظ :

```
Assign (f,'wajdy.txt');
```

```
Append (f);
```

اي لاضافه المزيد من البيانات الى داخل الملف النصي wajdy.txt تستخدم باسكال ذاكره داخلية Buffer لحفظ المعلومات النصيه ، والـ buffer هو مكان مؤقت عاده يكون مصفوفه او مؤشر في موقع ما في الذاكره .

يستخدم الـ buffer لتسريع او cache عمليه القراءه او الكتابه لاي ملف ، حتى الملفات النصيه ، والـ Buffer مكان له كميته او مساحه معينه لكنه يكفي للعمليات في باسكال ، اما في حاله امتلاء الـ buffer تقوم باسكال بعملية (حفظ) flush المحتويات الى القرص .

في بعض الاحيان تحتاج الى التأكد من ان الـ buffer لا يتجاوز حده الاقصى مهلا ، انت قلت انه كافي للعمليات في باسكال ؟؟ نعم لكن ماذا لو انقطت الكهرباء او انغلق الجهاز لاي سبب من الاسباب في هذه الحاله فإن كل محتويات الـ buffer سوف تضيع هباءا منثورا ، لذلك يجب حفظ flush محتويات القرص بشكل يدوي لانه يحفظ اول بأول ، انت لا تريد مستخدم برنامج يقول ااه ااه ااه لقد فقدت بياناتي 😊

حفظ محتويات الـ Buffer الى القرص بشكل يدوي يكون باضافه الجمله الاتيه /

Flush (f);

ويمكن كتابته في مكان بعد تعيين الملف الى اغلاق الملف ، لكن عاده يفضل قبل اغلاق الملف .

هذا بالنسبه الى الملفات النصيه ، اما الملفات الثنائيه فالامر يختلف (وسوف نتكلم عنه بعد قليل) .

بعض الاحيان في حاله قراءه او اضافته معلومات لملف غير موجود ، باسكال يصرخ "Error" ثم يقوم بالخروج من البرنامج ، والرجوع الى

شاشه العمل ، بدون اخطار للمستخدم واعطائه معلومات عن الخطأ ونوعه وما الى ذلك .

لذلك قدمت باسكال روتيناً لمعالجة الاخطاء error-handling ، دعنا نشاهده عن قرب //

{I-} → لجعل باسكال صامته عند اكتشاف الاخطاء  
: → العمليات على الملف  
{I+} → لجعل باسكال تقرر نوع الخطأ اذا وجد

واكتشاف الاخطاء يكون باضافه هذه الداله IOResult ، اذا كانت IOresult تساوي صفر 0 معناه انه لا يوجد خطأ .

// مثال

**Program** example (input,output);

**Uses** wincrt;

**Var**

F:text;

S:string;

**Begin**

Write ('Enter the file name: ');

Readln(s);

Assign (f, s);

{I-}

```

Reset (f);
{$I+}
If ioresult <> 0 then
    Begin
        Writeln ('error while reading file', s);
        Halt;
    End;
While not eof do
    Begin
        Readln (f, s);
        Writeln(s);
    End;
    Flush (f);
    Close (f);
End.

```

{SI+} و {SI-} تكون عادة بين فتح الملف ، كما في المثال السابق ، قم بتجربه البرنامج والتأكد من معرفته .

ان تضمين الـ IOResult يتسبب في اعاده رمز الخطأ ، من المفترض ان تقوم بحفظ محتوياته اولاً ثم اكتشاف الخطأ ، نفرض n متغير من النوع الحقيقي integer.

الان نرجع الى المثال السابق ، سأكتب ما بعد الـ Begin للبرنامج الرئيسي

```

:
{$I-}
Reset (F);
{$I+}
n:=IOResult;
If n<>0 then
Begin
  Writeln ('Error encountered in reading file ', s);
  Case n of
    2: writeln ('File not found');
    3: writeln ('Path not found');
    4: writeln ('too many open files');
    5: writeln ('File access denied');
    100: writeln ('Disk read error');
    101: writeln ('Disk write error');
    150: writeln ('Disk is writing protected');
    152: writeln ('Drive is not ready');
    154: writeln ('CRC error in data');
    156: writeln ('Disk seeks error');
    157: writeln ('Unknown media type');
    162: writeln ('Hardware failure');
  Else writeln ('various error');
End;
Halt;
End;

```

هكذا تستطيع اكتشاف اغلب الاخطاء الشائعه ، وايضا تخبر مستخدم برنامجك بالخطأ عارضا عليه رساله الخطأ ، يجب عليك ان تكتب برامج بهذه الطريقة اثناء التعامل مع الملفات ، انا اعلم انها مضجره قليلا لكن انت تريد ان تكون برامجك سهله التعامل مع المستخدم .

## الملفات النصيه الثائيه Binary File

يوجد نوعين من الملفات الثائيه هما /

\* الملفات المطبوعه Typed File

\* الملفات غير المطبوعه Untyped File

الملفات المطبوعه Typed File تعني ان الملف يحتوي صيغه واحده في كافه انحاء محتوياتها ، هذا النوع يتضمن قواعد البيانات لان كل محتوياتها سجلات بيانات (سجل ملفات)

الملفات الغير مطبوعه Untyped File تعني ان الملف لا يحتوي على صيغه واحده ، هذا النوع يحتوي على معلومات اضافيه قد تكون تركيبا مختلفا للسجلات .

اولا : Typed File

لنفترض بأنك عرفت سجل بهذا الشكل

**Type**

**Data = record**

Name: **string** [10];

Age: integer;

Tel: integer;

**End;**

Typed file لهذا السجل (السابق) سوف يكون بالشكل

**Var**

F: **file of data;**

الان الخطوات التي سنتبعها في الملفات المطبوعه Typed File تشبه الخطوات التي في الملفات النصيه text file .

نقوم باسناد المتغير F باسم الملف وذلك بالعباره assign

افتح الملف reset ، لانشاء ملف جديد rewrite

يجب تغير writeln و readln الى write و read

اغلق الملف وذلك بالعباره close

كل طرق معالجه الاخطاء Error Handling و Ioresult يستخدم كما هو ، ولا داعي لاعاده ذكره مره اخرى .

الفرق بين الملفات النصيه text file والملفات المطبوعه typed file هو انك اذا قمت بفتح ملف باستخدام reset هذا لا يعني انك تستطيع القراءه منه فقط (كما في الملفات النصيه) . بل يمكن اضافته المزيد وايضا امكانيه تعديله .

**Program example (input, output);**

**Uses** wincrt;

**Type**

**Data=record**

    Name: **string**;

    Age: integer;

**End;**

**Var**

F: **file of data**;

D: data;

C: char;

S: string;

**Begin**

    Write ('Input File name: ');

    Readln(s);

    Assign (f, s);

    Rewrite (f);

**Repeat**

        Clrscr;

        Write ('Name= '); readln (d.name);

        Write ('Age = '); readln (d.age);

        Write (f, d);

        Write ('Input New Data Y/N ');

**Repeat**

            C: = upcase (readkey);



```
    Until c in ['Y','N'];  
    Write(c);  
    Until c='N';  
    Close (f);  
End.
```

لم يكتمل بعد .....

## اليوم الثاني عشر

### الكلمات المحجوزه & الدوال الحسابيه

بعد انهائك للايام السابقه ، اصبحت مبرمجا في باسكال اليس كذلك  
(اتمى ذلك) .

لكن هناك العديد من المواضيع التى لم اضعها لسبب او لآخر على  
العموم في هذا اليوم سوف نتعلم الكلمات المحجوزه في باسكال ، لكنى  
اعتقد انك تعرفها جميعا .

ايضا سوف نأخذ بعض الدوال الحسابيه التى تفيدك في حاله اردت عمل  
برامج رياضيه .

## الكلمات المحجوزه :

=====

<b>AND</b>	<b>DOWNTO</b>	<b>IF</b>	<b>PACKED</b>	<b>TO</b>
<b>ARRAY</b>	<b>ELSE</b>	<b>IN</b>	<b>PROCEDURE</b>	<b>TYPE</b>
<b>BEGIN</b>	<b>END</b>	<b>LABEL</b>	<b>PROGRAM</b>	<b>UNTIL</b>
<b>CASE</b>	<b>FILE</b>	<b>MOD</b>	<b>RECORD</b>	<b>USES</b>
<b>CONST</b>	<b>FOR</b>	<b>NOT</b>	<b>REPEAT</b>	<b>VAR</b>
<b>DIV</b>	<b>FUNCTION</b>	<b>OF</b>	<b>SET</b>	<b>WHILE</b>
<b>DO</b>	<b>GOTO</b>	<b>OR</b>	<b>THEN</b>	<b>WITH</b>

## الدوال الحسابيه :

=====

النتاج	مثال	الوظيفه	الداله
10	ABS(-10);	تعطي القيمه المطلقه للعدد	ABS
25	SQR(5);	تعطي مربع العدد المدخل	SQR
5	SQRT(25);	تعطي الجذر التربيعي للعدد	SQRT
2	TRUNC(2.55);	تحذف الجزء العشري من العدد	TRUNC
6	ROUND(5.91);	تقرب العدد لاقرب عدد صحيح	ROUND

# النهايه

=====

المراجع التي استفدت منها في هذا الكتاب :

الكتب الاجنبيه

للکاتب Marco cantu  
للکاتب Rob miles  
للکاتب M.F.Somers  
لصاحب اللغة Wirth  
للکاتب Roby Joehanes

EssentialPascal  
Introduction to Pascal  
Learning PASCAL  
PascalRevisedReport  
The Basic of Pascal

الكتب العربيه

معايير في تقييم لغات البرمجه  
الدروس المترجمه في باسكال  
للکاتب عروه  
للکاتب MaaSTaaR

بالنسبه للمواقع العربيه والاجنبيه

الموسوعه العربيه  
الفريق العربي للبرمجه  
والعديد من الـ Tutorial المنتشره في انحاء الانترنت ولا يسع المكان  
هنا لنشرها ، علما بأن اغلبها بالانجليزيه.  
الموقع [www.c4arab.com](http://www.c4arab.com)  
الموقع [www.arabteam2000.com](http://www.arabteam2000.com)

## تم بحمد الله

نسأل الله تبارك وتعالى ، ان يكون هذا العمل خالصا لوجهه  
الكريم وأن يعيننا على تعلمه وتبليغه بإذنه ، إنه ولي ذلك  
والقادر عليه

وأخيرا ، إن كان من صواب فمن الله تعالى ، وإن كان من خطأ  
فمن انفسنا والشيطان .

وصلى اللهم وسلم وبارك على نبينا محمد وعلى اله وصحبه  
اجمعين .

واخر دعوانا أن الحمد لله رب العالمين .

والسلام عليكم ورحمه الله وبركاته .

جميع الحقوق محفوظة لـ وجدي عصام

**All Copy Righth is reseved to wajdy essam**

تم البدايه في هذا الكتاب بتاريخ **3/7/2005** والانتهاء منه بتاريخ **3/9/2005**