في هذا الفصل سنتعرف على مقدمة عامة عن البرمجة من خلال تعريف البرمجيات وخصائصها وأنواع الأخطاء فيها والعناصر المكونة للغة البرمجة. كما سنتعرف على مقدمة عن أنواع البيانات والتعليمات والعمليات التي تستخدم في البرمجة وتعريف المتغيرات والثوابت والقواعد العامة المتعلقة بأنواع البيانات وأسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطةوذلك بالإضافة إلى جملة من الأمثلة والتمارين التطبيقية لمحتويات الفصل

1.1) مقدمه عن البرمجيات INTRODUCTION TO SOFTWARE

في هذا الفصل سنقوم بعرض مقدمة عامة عن البرمجة من خلال تعريف البرمجة وخصائصها وأنواع الأخطاء بها والعناصر المكونة للغة البرمجة وذلك بالإضافة إلى مقدمة عامة عن أنواع البيانات والقواعد المتعلقة بها وأنواع التعليمات والعمليات المستخدمة في البرمجة وتعريف المتغيرات والثوابت وأسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطة

1.1.1)تعريف البرمجيات Software definition

هي مجموعة من التعليمات (instructions) التي يتم استخدامها في بناء البرنامج وتقوم المكونات المادية للحاسب (الذاكرة – المعالج – وحدات الإدخال والإخراج) بتنفيذها لتؤدي مهام ووظائف معينه وتكتب هذه التعليمات بأحد اللغات المستخدمة في البرمجة مثل python, Fortran, Basic, Cpp , and Java

1.1.2)خصائص البرمجيات Characteristics of Software

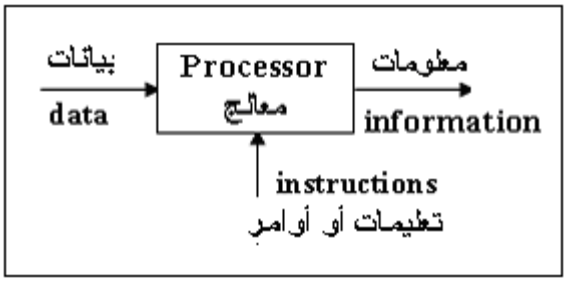
تتميز البرمجيات بالخصائص التالية:

1) لها بداية
2) لها نهاية
3) لها جسم
4) تحقق التسلسل المنطقي للتعليمات
5) الدقة في التصميم
6) لها مدخلات (data – بيانات) ومخرجات (information - معلومات)

المعلومات (information) هي بيانات تمت معالجتها وأحيانا تسمى (processed data).
الشكل رقم 1 يوضح عملية الحصول على المعلومات من خلال معالجة البيانات بواسطة التعليمات (instructions)

1.1.3)أنواع الأخطاء البرمجية Types Of Software Errors

تنقسم الأخطاء البرمجية إلى أخطاء لغوية (syntax errors) وأخطاء منطقية (logical errors)



شكل 1: عملية الحصول على المعلومات من خلال معالجة البيانات بواسطة التعليمات

1) الأخطاء اللغوية (syntax errors):

هي أخطاء تعتمد علي اللغة المستخدمة وقواعدها ويمكن كشف هذه الأخطاء بواسطة استخدام المعالجات (compilers) أو المجمعات (assemblers) أو المفسرات (interpreters) .
فعلى سبيل المثال لغة C تشترط وضع (;) في نهاية الجملة على النحو التالي:

A = B + C ;

فإذا لم نضع هذه العلامة في نهاية الجملة فسيكون هناك خطأ لغوي و في العبارة التالية أيضا يوجد خطأ لغوى حيث وجود علامتي الجمع والطرح بصورة متتالية وهو منافي لقواعد العمليات الحسابية

A = A+-C ;

2) الأخطاء المنطقية (logical errors) :

هى أخطاء في طريقه كتابة المعادلات وإجراء الحسابات ويمكن اكتشافها باستخدام طرق المحاكاة (Simulation) فعلى سبيل المثال يريد المبرمج أن يوجد مجموع قيمتي المتغيرين A and B ولكنه استخدم العبارة:

C = A – B

بدلا من العبارة

C = A + B

لن يخبره المترجم عن وجود خطأ وبالتالي يجب استخدم طرق المحاكاة (Simulation) لاكتشاف هذا الخطأ.

بصورة مبسطة عملية المحاكاة (Simulation) تعني إدخال بيانات محددة معلوم مسبقا قيمة ناتج البرنامج لها فإذا كانت مخرجات البرنامج تساوي الناتج المعرف مسبقا كان البرنامج خاليا من الأخطاء المنطقية وإذا كان هناك اختلاف بين الناتج المعرف مسبقا ومخرجات البرنامج يكون هناك خطأ منطقي وبالتالي يجب مراجعة عبارات وخطوات البرنامج وإجراء المحاكاة مرة أخري حتى يتم التأكد من أن البرنامج خاليا من الأخطاء المنطقية.

1.1.4)مكونات لغة البرمجة Software Compositions

تتكون لغة البرمجة من مجموعة من القواعد والمصطلحات يستخدمها المبرمج لكتابة برنامج معين يفهمه الحاسب حسب نوعه ويتم تحويل البرنامج طبقاً للغة المكتوب بها إلي لغة الآلة (machine language) بواسطة المعالجات compilers أو المجمعات assemblers

1.1.5)العناصر الأساسية للغة البرمجة

The Main Element of Software Languages

تتكون لغات البرمجة من العناصر التالية:

a) مجموعة من الرموز الأساسية

1) حروف لاتينية : A, B, C, ….., X, Y, Z

2) أرقام عربية : 0,1,2,3,4,…… 9

3) رموز خاصة : $, % , / , * , - , +

b) الكلمات

وهي نوعان:

1) كلمات محجوزة (reserved words): هي كلمات لها معني خاص بالحاسب وتقوم بتنفيذ مهام محددة ويوجد قيود عليها حسب لغة المبرمجة المستخدمة مثل:

Read, Write, Add, for, and if

2) كلمات يختارها المستخدم أو المبرمج(user defined words) : هي كلمات يختارها المبرمج لتمثيل أسماء المتغيرات والثوابت ولا يوجد قيود على استخدامها مثل :

Alpha, A, ABD, sum, etc.

c) مجموعة التعليمات Instructions set

هي مجموعة من الرموز أو الكلمات الخاصة بلغة البرمجة لتنفيذ عملية أو مهام أو أمر معين وتنقسم إلى ثلاثة أنواع:

1) تعليمات خاصة بالذاكرة (memory reference instructions)

وتقوم غالبا بتنفيذ العمليات الحسابية أو المنطقية مثل: .ADD, SUB, *, /, +, etc. فعلى سبيل المثال العبارة

C = A + B

تعنى جمع محتوى الذاكرة الموجودة في العنوان A مع محتوى الذاكرة الموجود في العنوان B ووضع الناتج في العنوان C داخل الذاكرة

2) تعليمات إدخال وإخراج (Input/ output instructions)

هي تعليمات لإدخال البيانات وإخراج المعلومات مثل: Write and Read. فعلى سبيل المثال:

Read (X, Y, Z)

Write (A, B, C)

العبارة الأولى تعني إدخال قيم المتغيرات (X, Y and Z) من خلال وحدة الإدخال والعبارة الثانية تعني إخراج قيم المتغيرات (A, B and C) إلى وحدة الإخراج

3) تعليمات تحكم (Control instructions):

تشمل التعليمات التي تؤدي إلي تسلسل تنفيذ البرنامج مثل

Goto, While, Do-while, switch-case, for …etc.

عندما يفهم الحاسب هذه التعليمات تقوم وحده التحكم بالجهاز (control unit) بإرسال الإشارات أو أوامر التحكم اللازمة إلي الوحدات المادية المعنية بذلك لتنفيذ هذه التعليمات

يتم تنفيذ كل تعليمة من خلال ثلاث مراحل أساسية :

1) مرحلة استحضار العبارة التي تحتوى على التعليمة من ذاكرة الحاسب (Fetch)
2) مرحلة تفسير هذه التعليمة لفهم مدلولها (decoding)
3) مرحلة تنفيذ هذه التعليمة (execution)

مثال: 1

A = A + B

لتنفيذ العبارة السابقة يتم إتباع التالي

1) الذهاب إلى ذاكرة الحاسب لاستحضار العبارة السابقة حيث أنها تكون مخزنة مسبقا في ذاكرة الحاسب بلغة الآلة (machine language)
2) تفسير التعليمة (instruction) الموجودة بالعبارة وإجراء التالي:

- صدور أمر بواسطة وحدة التحكم بالحاسب (control unit) للذهاب إلي الذاكرة في العنوان A لمعرفة محتواه وإرساله إلى وحدة العمليات الحسابية والمنطقية (Arithmetic and logic unit – ALU)
- صدور أمر بواسطة وحدة التحكم بالحاسب (control unit) للذهاب إلي الذاكرة في العنوان B لمعرفة محتواه وإرساله إلى وحدة العمليات الحسابية والمنطقية (ALU)

3) صدور أمر لوحدة العمليات الحسابية والمنطقية (ALU) لإجراء عملية الجمع لمحتوى العنوان A ومحتوى العنوان B

4) في النهاية صدور أمر آخر بواسطة وحدة التحكم لتخرين المجموع في العنوان A

الشكل رقم 2 يوضح محتوى ذاكرة الحاسب قبل وبعد تنفيذ العبارة :A = A + B

	قبل تنفيذ العبارة
A	$(7)_{10}=(0111)_2$
	.
	.
	.
B	$(10)_{10}=(1010)_2$

	بعد تنفيذ العبارة
A	$(17)_{10}=(10001)_2$
	.
	.
	.
B	$(10)_{10}=(1010)_2$

شكل 2: محتوى ذاكرة الحاسب قبل وبعد تنفيذ العبارة :A = A + B

d) قواعد اللغة:

هي مجموعة من القواعد والقيود التي يجب الالتزام بها عند كتابه البرامج وعند مخالفه هذه القواعد والقيود يعطي البرنامج أثناء مرحله الترجمة أخطاء لغوية (Syntax errors)

1.2) المصطلحات الأساسية للغات البرمجة

قبل البدء في معرفة المفاهيم الأساسية للبرمجة يجب أن نتناول معنى المصطلحات الرئيسية للغات البرمجة بالشرح والتفصيل

1.2.1)البيانات Data

هي مدخلات الحاسب وتعتبر هي المادة الخام للبرنامج والتي لا تعطي معني بذاتها

أنواع البيانات Data types

البيانات تستخدم أنواع مختلفة يجب أن تحدد بدقة قبل استخدامها داخل البرنامج. الأنواع الأساسية للبيانات هي:

1) بيانات عدديه (Numeric data):

هي الأرقام الصحيحة (integer) والحقيقية (real) والثنائية (binary) مثل:

$(123)_{10}$, $(12.56)_{10}$, and $(1011011)_2$

البيانات الثنائية تمثل القيمة المناظرة للبيانات داخل الذاكرة

2) بيانات حرفيه (Character data):

البيانات الحرفية character data تتكون من جميع الأرقام (9 ,… ,2 ,1,0) والحروف الهجائية الصغيرة lower-case letters والحروف الهجائية الكبيرة upper-case letters والحروف الخاصة (&, $, #,..etc) والتي تكون في مجملها 256 حرفا. البيانات الحرفية تمثل داخل برامج الحاسب بوضعها بين العلامتين (' ') فعلى سبيل المثال البيانات التالية تمثل character data :

'9', 'a', 'A', and '%'

3) كلمات (String data)

هي مجموعه من الحروف تشكل كلمات. String data تمثل داخل برامج الحاسب بوضعها بين العلامتين (" ") فعلى سبيل المثال البيانات التالية تمثل string data :

“abck”, "Ahmed”, and "1234"

4) بيانات منطقية Logical data(Boolean data) :

هي بيانات تأخذ احد القيمتين (False or True). بمكن استخدام T أو yes بدلا من القيمة true و F أو no بدلا من القيمة false. تستخدم البيانات المنطقية في التركيبات الشرطية

5) بيانات مركبة (Composite data):

تنقسم البيانات المركبة إلى نوعين رئيسيين هما المصفوفات (arrays) والسجلات /التركيبات (records/structures)

a) المصفوفات (array):

هي مجموعة من البيانات من نفس النوع

مثل : مصفوفة الأرقام الصحيحة {34 ,15 ,12} التي تحتوي على ثلاثة عناصر من البيانات الصحيحة و مصفوفة الأرقام الحقيقية {1.6 ,12.4 ,12.6} التي تحتوي على ثلاثة عناصر من البيانات الحقيقية

b) السجلات (records) أو التركيبات (structures) :

هي مجموعة من البيانات المختلفة في النوع مثل سجلات الموظفين حيث أن كل سجل يحتوي على اسم الموظف (كلمات string) ورقم الموظف (رقم صحيح integer) وراتبه الشهري (رقم حقيقي real). فعلى سبيل المثال

("Ahemd", 12450, 'B', 2020.5)

البيانات السابقة تكون سجل أو تركيب يحتوي على أربعة عناصر. العنصر الأول "Ahmed" (string data) يمثل اسم الموظف والعنصر الثاني 12450 (integer data) يمثل رقم الموظف والعنصر الثالث 'B'(character data) يمثل درجة الموظف الوظيفية والعنصر الرابع 2020.5(real data) يمثل راتب الموظف الشهري

1.2.2)التعليمات Instructions

تنقسم التعليمات إلى تعليمات خاصة بالذاكرة وتعليمات خاصة بوحدات الإدخال والإخراج وتعليمات خاصة بالتحكم وهو ما تم تناوله سابقا

1.2.3)المعلومات Information

ناتج معالجة البيانات (Processed data) بواسطة إجراء التعليمات أو العمليات على البيانات تعطي قيم لها معني بذاتها تسمى معلومات . يمكن تعريف المعلومات أيضا بأنها هي مخرجات البرنامج التي تظهر في صورة تقارير أو جداول أو رسوم .

1.2.4)العوامل Operators

يجب إخبار الحاسب عن كيفية إجراء المعالجة للبيانات وذلك من خلال استخدام مجموعة من العوامل operators التي تعتبر data connectors في التعبيرات expressions والمعادلات equations . العوامل operators هي إشارات signs أو رموز symbols تخبر الحاسب عن كيفية معالجة البيانات وأيضا تخبر الحاسب عن نوع المعالجات (عمليات حسابية أو منطقية) المراد إجراؤها على البيانات وذلك بالإضافة إلى أنواع المتغيرات والثوابت التي يمكن استخدامها مع هذه العوامل .

العوامل Operators والمتغيرات variables والثوابت constants تجمع سويا لتكوين التعبيرات expressions أو المعادلات equations

أنواع العوامل المستخدمة في الحسابات وحل المسائل هي : معاملات حسابية ومعاملات منطقية ومعاملات علاقة وهو ما سنتناوله لاحقا.

1.2.5)المعاملات والمحصلات Operands and resultants

المعاملات operands هي البيانات التي توصل وتعالج بواسطة العوامل operators . المحصلات resultants هي الناتج أو المخرجات بعد إجراء المعالجات اللازمة على المعاملات فعلى سبيل المثال في التعبير 15 + 12 : + تمثل العامل operators والبيانات الرقمية 12 and 15 هما العوامل operands وناتج المعالجة 27 هو المحصلة resultant

المعاملات operands يمكن أن تكون متغيرات variables أو ثوابت constants. نوع البيانات للمعاملات تعتمد على نوع العوامل operators . نوع المحصلة resultant تعتمد على نوع كل من العوامل والمعاملات

1.2.6)التعبيرات Expressions والمعادلات Equations

التعبيرات والمعادلات تمثل تعليمات instruction تستخدم للوصول إلى حل المشكلة

1.2.7)الدوال Functions

الدوال Functions هي مجموعة من التعليمات instructions تؤدي وظيفة أو عدة وظائف محددة

1.3) المتغيرات والثوابت VARIABLES AND CONSTANTS

الحاسبات تستخدم المتغيرات والثوابت في حل المسائل. المتغيرات والثوابت هما بيانات تستخدم في معالجات الحاسب. بصورة أوضح البيانات التي نتعامل بها من خلال البرامج تمثل قيمة متغير أو قيمة ثابت

1.3.1)المتغيرات Variables

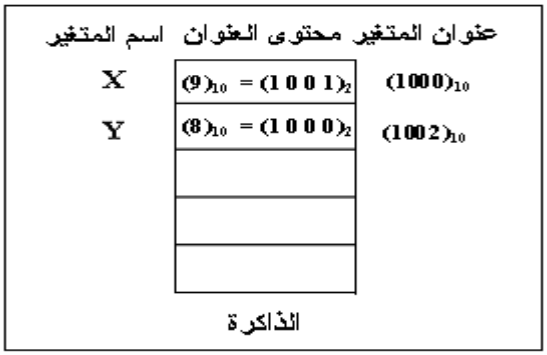
هي بيانات تتغير قيمتها أثناء تنفيذ البرنامج. يمكن تسمية هذه المتغيرات بحروف أو كلمات. من المهم جدا فهم الفارق بين اسم المتغير وقيمته. اسم المتغير يمثل علامة label يستخدمها الحاسب لإيجاد الوضع الصحيح للمتغير داخل ذاكرة الحاسب وقيمة المتغير تمثل محتوى هذا الموضع.

الحاسب يستخدم اسم المتغير لإيجاد موضعه داخل الذاكرة ويستخدم قيمة المتغير لإجراء المعالجات والعمليات. حيث أن اسم المتغير يمثل موضع تخزين المتغير فبالتالي يمكن القول أن اسم المتغير يمثل عنوان address تخزين قيمة البيانات التي تمثل محتوي هذا العنوان وحيث أن الموضع داخل الذاكرة معرض لتغير محتواه فان المتغيرات هي بيانات تتغير أثناء تنفيذ البرنامج. فعلى سبيل المثال

$$X = (9)_{10} = (1\ 0\ 0\ 1)_2$$

$$Y = (8)_{10} = (1\ 0\ 0\ 0)_2$$

المتغير X يمثل بالعنوان $\mathbf{(1000)_{10}}$ في الذاكرة والذي يحتوي على القيمة $(9)_{10}$ في النظام العشري أو القيمة $(1001)_2$ في النظام الثنائي. المتغير Y يمثل بالعنوان $\mathbf{(1002)_{10}}$ في الذاكرة والذي يحتوي على القيمة $(8)_{10}$ في النظام العشري أو القيمة $(1000)_2$ في النظام الثنائي. الشكل رقم 3 يمثل هيكلا لجزء من ذاكرة الحاسب يتضمن محتوى وعنوان المتغيرين X and Y



شكل 3: هيكلا لجزء من ذاكرة الحاسب يتضمن محتوى وعنوان المتغيرين X and Y

الذي تمثل قيمة المتغير (Data)عند الإعلان عن المتغيرات يجب تحديد نوع البيانات

أمثلة:

variable X, Y: integer

الإعلان عن المتغيرين X, and Y من النوع integer

variable A1, B2: real

الإعلان عن المتغيرين A1, and B2 من النوع real

variable ch: character

الإعلان عن المتغير ch من النوع character

variable AM[20]: character

variable AM: string

العبارة الأولى تمثل الإعلان عن المتغير AM من النوع string (سلسلة حروف يحتوي على 20 حرف). يمكن استخدام العبارة الثانية في الإعلان عن المتغير AM من النوع string

variable BOL1: Boolean

الإعلان عن المتغير BOL1 من النوع Boolean

يمكن إعطاء قيم أولية للمتغيرات أثناء الإعلان عنها على النحو التالي

variable A = 20, B: integer

تهيئة المتغير A بالقيمة الأولية 20

variable ch ='K', ch1: character

تهيئة المتغير ch بالقيمة الأولية 'K' . لاحظ أننا وضعنا الحرف بين العلامتين ' ' للدلالة على وجود قيمة من النوع character

variable F1 = 12.5, F2: real

تهيئة المتغير F1 بالقيمة الأولية 12.5

variable M1=True: Boolean

تهيئة المتغير M1 بالقيمة الأولية True

variable X1[5] ={12, 34, 5, 13, 42}: integer

تهيئة المتغير X1 بالقيمة الأولية 12, 34, 5, 13, and 42. لاحظ أننا وضعنا الأرقام الصحيحة بين العلامتين { }للدلالة على وجود مصفوفة عناصرها من النوع integer ولاحظ أيضا أننا وضعنا الرقم 5 بين القوسين [ ] لتحديد عدد عناصر المصفوفة

variable X2[3] = { 12.6, 10.4, 6.0, 4.2, 19.3} : real

تهيئة المتغير X2 بالقيمة الأولية 12.6, 10.4, 6.0, 4.2, and 19.3. نلاحظ أننا وضعنا الأرقام العشرية بين العلامتين { }للدلالة على وجود مصفوفة عناصرها من النوع real ولاحظ أيضا أننا وضعنا الرقم 3 بين القوسين [ ] لتحديد عدد عناصر المصفوفة

variable X3[8] = “Computer” : character

تهيئة المتغير X3 بالقيمة الأولية Computer. نلاحظ أننا وضعنا الأحرف بين العلامتين “ ”للدلالة على وجود مصفوفة من الحروف و لاحظ أيضا أننا وضعنا الرقم 8 بين القوسين [ ] لتحديد عدد عناصر المصفوفة

variable X4[8] = {‘C’, ‘o’, ‘m’,’p’,’u’,’t’,’e’,’r’} : character

يمكن عمل تهيئة بصورة أخرى لمصفوفة الحروف وذلك بوضع الأحرف بين العلامتين { } للدلالة على وجود مصفوفة ووضع كل حرف بين العلامتين ‘ ’

يمكن إسناد قيمة إلى المتغير خارج عبارة الإعلان عن المتغير وذلك من خلال إدراج عبارات الإسناد التالية داخل جسم البرنامج

A = 20

ch =’K’

F1 = 12.5

M1=True

1.3.2)الثوابت Constants

هي بيانات لا تتغير قيمتها أثناء تنفيذ البرنامج

أمثلة:

constant A3=20: integer

A3 مقدار ثابت من النوع integer قيمته 20

constant AA = 12.4 real

AA مقدار ثابت من النوع real قيمته12.5

constant name = “Ali”: string,

Or

constant name[5] = “Ali”: character

name مقدار ثابت من النوع string قيمته ”Ali“

عند وجود الثابت داخل جسم البرنامج يقوم المترجم (compiler) بالتعويض عن قيمة الثابت بالقيمة المحددة عند الإعلان عنه. فعلى سبيل المثال في العبارة

Y = A3 + 10

يقوم المترجم بالتعويض عن الثابت A3 بالقيمة 20 المعرفة مسبقا في العبارة

: integerconstant A3 = 20

وبالتالي تكون قيمة المتغير Y تساوي 30

1.4) عمليات البرمجة Software Operations

يوجد ثلاث عمليات رئيسية يتم استخدامها في البرمجة وهي:

a) عمليات حسابية Arithmetic operation
b) عمليات منطقية Logical operations
c) عمليات العلاقة Relational operations

1.4.1)العمليات الحسابية Arithmetic operations

الجدول رقم 1 يوضح العمليات الحسابية وعامل (operator) كل عملية

جدول 1: العمليات الحسابية وعامل (operator) كل عملية

العملية	operatorالعامل
الجمع	+

الضرب	*
الطرح	-
القسمة	/
باقي القسمة	MOD% أو
الأس	** أو ^

هنـاك عوامـل (operators) حسـابية أخـري تسـمى عوامـل التوظيـف (assignment operators) تختص باللغة المستخدمة وخصائصها . الجدول رقم 2 يوضح عوامل التوظيف (assignment operators) في لغة C والمعنى المكافئ لكل عامل. معاملات العمليات الحسابية والمحصلة (operands and resultants) يكون من النوع numeric data (أرقام صحيحة integer numbers أو أرقام حقيقية real numbers) . فعلى سبيل المثال إذا كان :A = 30 and B = 4 فان :

C1 = A + B ⟶ C1 = 34

C2 = A – B ⟶ C2 = 26

C3 = A / B ⟶ C3 = 7.5

C4 = A * B ⟶ C4 = 120

C5 = A Mod B ⟶ C5 = 2

حيث أن A, and B هي أسماء متغيرات تمثل معاملات (operands) تحتوي على بيانات صحيحة (integer numbers) بينما C1, C2, C3, C4, and C5 هي أسماء متغيرات تمثل محصلات (resultants) تحتوي على بيانات صحيحة أو عشرية حسب نوع العامل المستخدم في العملية

جدول 2: عوامل التوظيف (assignment operators)

في لغة C والمعنى المكافئ لكل عامل

عوامل التوظيف والمعاملات	المعنى المكافئ
X++	استخدم القيمة الموجودة أولا للمتغير X ثم أضف 1إلى هذه القيمة بعد ذلك
X—	استخدم القيمة الموجودة أولا للمتغير X ثم اطرح 1 من هذه القيمة بعد ذلك
++X	زد قيمة المتغير X بمقدار 1 أولا ثم استخدم القيمة الجديدة بعد ذلك
--X	اطرح 1 من قيمة المتغير X ثم استخدم القيمة الجديدة بعد ذلك
-X	اعكس إشارة المتغير X
X=Y	قيمة المتغيرX تساوي قيمة المتغير Y
X*=Y	تكافئ العبارة X=X*Y
X/=Y	تكافئ العبارة X=X/Y
X%=Y	تكافئ العبارةX=X%Y
X+=Y	تكافئ العبارة X=X+Y
X-=Y	تكافئ العبارةX=X-Y

1.4.2)عمليات العلاقة Relational operations

الجدول رقم 3 يوضح عمليات العلاقة وعامل (operator) كل عملية

المعاملات (operands) تكون بيانات عددية: صحيحة (integer) أو حقيقية (real).

ناتج عمليات العلاقة يكون بيانات من النوع (False or True)data (Boolean) logical .

فعلى سبيل المثال إذا كان A = 10 and B = 5 فان :

C1 = A > B ⟶ C1 = True (T)

C2 = A < B ⟶ C2 = False (F)

C3 = A ==B ⟶ C3 = False (F)

حيث أن A, and B هما أسماء متغيرات تمثل معاملات (operands) تحتوي على بيانات صحيحة. C1, C2, and C3 هي أسماء متغيرات تمثل محصلات (resultants) تحتوي على بيانات Boolean تأخذ قيم False (F) or True (T). إذا تحقق الشرط يكون ناتج العلاقة يساوي True (T) وإذا لم يتحقق الشرط يكون ناتج العلاقة False (F). تستخدم عمليات العلاقة في التركيبات الشرطية

جدول 3: عمليات العلاقة وعامل (operator) كل عملية

العملية	العامل operator
يساوي	==
لا يساوي	=!
اقل من	>
اقل من أو يساوي	>=
اكبر من	<
اكبر من أو يساوي	<=

1.4.3)العمليات المنطقية Logical operations

الجدول رقم 4 يوضح العمليات المنطقية وعامل (operator) كل عملية. الجدول رقم 5 يوضح ناتج العمليات المنطقية. العوامل المنطقية logical operators تستخدم لربط تعبيرات العلاقة relational expressions مثل :

A > B && A < 9

بالإضافة إلى إجراء العمليات operations على البيانات المنطقية (logical (Boolean data

جدول 4: العمليات المنطقية وعامل (operator) كل عملية

العملية	العامل operator
NOT	!
AND	&&
OR	||

جدول 5 : ناتج العمليات المنطقية

A	B	A&&B	A||B	!A	!B
F	F	F	F	T	T
F	T	F	T	T	F
T	F	F	T	F	T
T	T	T	T	F	F

المعاملات (operands) تكون بيانات منطقية logical (Boolean) data(T or F). ناتج العملية المنطقية يكون من النوع logical (Boolean) data(F or T). فعلى سبيل المثال إذا كان A = T, and B = F فان:

C1 = A && B ⟶ C1 = F

C2 = A || B ⟶ C2 = T

C3 = !A ⟶ C3 = F

حيث أن A, B, C1, C2, and C3 هي أسماء متغيرات تمثل معاملات (operands) تحتوي على بيانات منطقية (T or F)

1.5) قواعد عامة متعلقة بأنواع البيانات

GENERAL RULES FOR DATA TYPES

1) تعريف البيانات قبل استخدامها:

قبل استخدام أي بيانات في البرنامج يجب تعريفها. يتم تعريف البيانات من ناحيتين

a) ناحية الغرض Data object:

- Variable
- Constant

b) ناحية النوع Data type

- integer
- real
- character
- string
- Boolean

2) في معظم لغات البرمجة لا يصح إجراء عمليات علي بيانات من أنواع مختلفة (different data types) فعلى سبيل المثال إذا تم الإعلان عن المتغيرات Y, F, X and K بالعبارات التالية

variable Y,F: integer

variable X: character

variable K: real

واستخدمت العبارة التالية

Y = X + F

فان هذا العبارة تكون (في معظم اللغات) خاطئة حيث أن المتغيرات من أنواع مختلفة (المتغير X من النوع character والمتغير F من النوع integer) .

بعض لغات البرمجة يمكن أن تعتبر هذه العبارة صحيحة على أساس أنه يتم تحويل البيانات الصحيحة إلى حقيقية أو العكس أو تحويل الحرف إلى قيمة ASCII code الخاص به تلقائيا

3) يجب أن تتناسب نوع العملية مع المعاملات (operands). فعلى سبيل المثال إذا تم الإعلان عن المتغيرات X1, X2, x3, A1, A2,and A3 بالعبارات التالية

variable X1, X2, X3: BOOL

constant A1, A2, A3: integer

واستخدمت العبارتين التاليتين:

X1 = X2 + X3

A1 = A2 && A3

فان العبارة الأولى تكون خاطئة حيث أنه لا يمكن إجراء العمليات الحسابية (الجمع) على متغيرات من النوع Boolean والعبارة الثانية تكون خاطئة أيضا حيث أنه لا يمكن إجراء العمليات المنطقية (&&)على متغيرات من النوع integer

4) لا يمكن خلط البيانات فعلى سبيل المثال لا يمكن وضع string data في موضع متغير داخل الذاكرة يكون معرفا على أن محتواه numeric data

5) كل نوع من البيانات يستخدم قائمة محددة من البيانات data set فعلى سبيل المثال integer data تستخدم جميع الأرقام الموجبة والسالبة و real data تستخدم الأرقام العشرية و character data تستخدم جميع الحروف والرموز الخاصة مع وضع الحرف بين العلامتين (' ') و Boolean data تستخدم أحد القيمتين true أو false . إذا تم استخدام بيانات خارج هذه القائمة يكون هناك خطأ

1.6) التعبيرات والمعادلات Expressions and Equations

التعبيرات expressions تعالج البيانات (operands) من خلال استخدام عوامل operators محددة. فعلى سبيل المثال إذا كان X معامل يمثل طول مستطيل والمعامل Y يمثل عرض المستطيل فان التعبير :

X * Y

يمثل مساحة المستطيل. ناتج التعبير لا يخزن في الذاكرة

المعادلات equations تخزن ناتج التعبير في موضع محدد داخل الذاكرة من خلال استخدام عامل الإسناد (=)assignment operator و بالتالي فان المعادلة تتكون من تعبير بالإضافة إلى عامل الإسناد. تعليمة الحاسب computer instruction تمثل بواسطة معادلة . فعلى سبيل المثال:

Z = X * Y

تمثل معادلة equation أو تعليمة instruction حيث يتم تخزين ناتج التعبير (X * Y) في موضع داخل الذاكرة محدد بالعنوان الذي يمثله المتغير Z

الجدول رقم 6 يوضح الفارق بين التعبيرات والمعادلات

جدول 6 : الفارق بين التعبيرات والمعادلات

Equationالمعادلة	Expressionالتعبير
D = A * C A and C معاملات operands تمثل بيانات من النوع numerical وتخزن المحصلة resultant(numerical data) في موضع داخل الذاكرة محدد بالعنوان الذي يمثله المتغيرD	**A * C** A and C معاملات operands تمثل بيانات من النوع numerical data ولا تخزن المحصلة resultant في الذاكرة
D = A > B A and B معاملات operands تمثل بيانات من النوع , numerical string or character data وتخزن المحصلة resultant(logical data) في موضع داخل الذاكرة محدد بالعنوان الذي يمثله المتغيرD	A > B A and B معاملات operands تمثل بيانات من النوع , numerical string or character data ولا تخزن المحصلة resultant في الذاكرة
A && B A and B معاملات operands تمثل بيانات من النوع logical data وتخزن المحصلة resultant(logical data) في موضع داخل الذاكرة محدد بالعنوان الذي يمثله المتغيرD	A && B A and B معاملات operands تمثل بيانات من النوع logical data ولا تخزن المحصلة resultant في الذاكرة

المعادلة equation تسمى أحيانا عبارة التوظيف assignment statement وذلك لأن قيمة التعبير على يسار عامل التوظيف يتم إسناده إلى المتغير على يسار عامل التوظيف

1.7) الأسبقيات PERIORITIES

في الجزء التالي سنناقش أسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطة

1.7.1)أسبقيات العمليات الحسابية Priorities of Arithmetic Operators

الجدول رقم 7 يوضح ترتيب أسبقيات العمليات الحسابية ونوع بيانات كل من operands and resultants

جدول 7 : ترتيب أسبقيات العمليات الحسابية

ونوع بيانات كل من operands and resultants

الترتيب	نوع بيانات operands	نوع بيانات resultant
1) الأقواس	Numerical	Numerical
2) الأس	Numerical	Numerical
3) باقي القسمة	Numerical	Numerical
4) الضرب والقسمة (من اليسار إلي اليمين)	Numerical	Numerical
5) الجمع والطرح (من اليسار إلي اليمين)	Numerical	Numerical

مثال: 2

أحسب ناتج المعادلات التالية مع توضيح خطوات التنفيذ

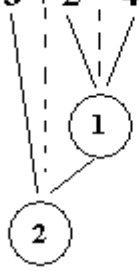
a) X = 3 + 2 * 4

b) Y = 2 * 6 + 3 – 2 * 4

c) Z = (2 * 5 ** 2) / 2 + 12

الحل:

a) X = 3 + 2 * 4

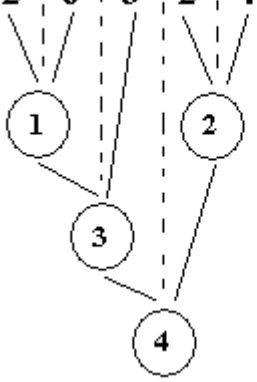


X = 3 + 2 * 4

X = 3 + 8

X = 11

b) Y = 2 * 6 + 3 – 2 * 4



Y = 2 * 6 + 3 – 2 * 4

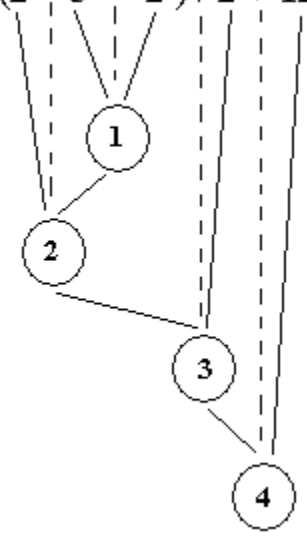
Y = 12 + 3 – 2 * 4

Y = 12 + 3 - 8

Y = 15 – 8

Y = 7

c) Z = (2 * 5 ** 2) / 2 + 12



Z = (2 * 5 ** 2) / 2 + 12

Z = (2 * 25) / 2 + 12

Z = 50 / 2 + 12

Z = 25 + 12

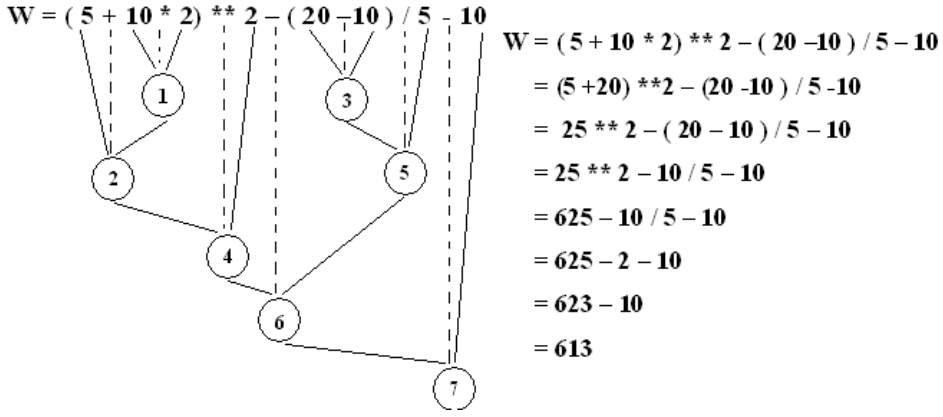
Z = 37

مثال: 3

إذا كان X = 5, Y = 10, Z = 20, أحسب ناتج المعادلة التالية مع توضيح خطوات التنفيذ

W = (X+Y *2) **2 – (Z- Y) /X – 10

الحل:



W = (5 + 10 * 2) ** 2 – (20 –10) / 5 – 10
= (5 +20) **2 – (20 -10) / 5 -10
= 25 ** 2 – (20 – 10) / 5 – 10
= 25 ** 2 – 10 / 5 – 10
= 625 – 10 / 5 – 10
= 625 – 2 – 10
= 623 – 10
= 613

1.7.2)أسبقيات عمليات العلاقة Priorities of Relational Operations

الجدول رقم 8 يوضح ترتيب أسبقيات عمليات العلاقة ونوع بيانات كل من operands and resultants

جدول 8 : ترتيب أسبقيات عمليات العلاقة

ونوع بيانات كل من operands and resultants

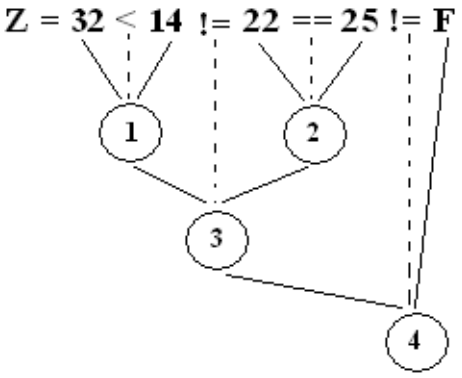
نوع بيانات resultant	نوع بيانات operands	الترتيب
logical	Numerical, strings or characters	==, <, <=, >, >=, <> (!=) من اليسار إلى اليمين

مثال: 4

احسب ناتج المعادلة التالية

Z = 32 <14 != 22 == 25 != F

الحل



$$Z = 32 < 14 != 22 == 25 != F$$
$$= F != 22 == 25 != F$$
$$= F != F !F$$
$$= F != F$$
$$= F$$

1.7.3)أسبقيات العمليات المنطقية Priorities Of Logical Operations

الجدول رقم 9 يوضح ترتيب أسبقيات العمليات المنطقية ونوع بيانات كل من operands and resultants

جدول 9 : ترتيب أسبقيات العمليات المنطقية

ونوع بيانات كل من operands and resultants

الترتيب	نوع بيانات operands	نوع بيانات resultant
1) الأقواس 2) NOT والتي تكافئ !	Logical Logical	Logical Logical

3) AND والتي تكافئ &&	Logical	Logical
4) OR والتي تكافئ ||	Logical	Logical

مثال: 5

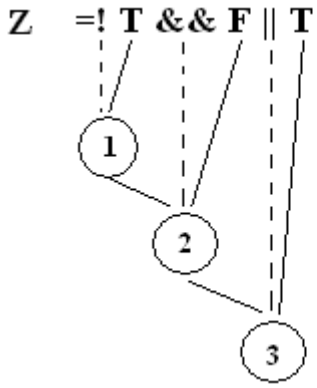
إذا كان A = T , B = F أحسب ناتج المعادلات التالية مع توضيح خطوات التنفيذ

a) Z = !A && B || A

b) Z = ! (A || B) && B

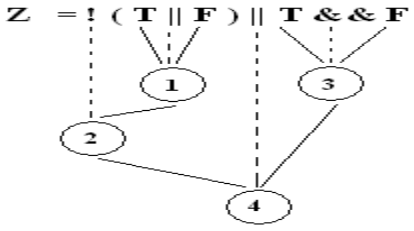
الحل:

a) Z = !A && B || A



Z = !A && B || A
=! T && F || T
= F && F || F
= F || T =
= T

b) Z = ! (A || B) || T && B



Z = ! (T || F) || T && F
= ! T || T && F
= F || T && F
= F || F
= F