برمجـــة 1

-

مبادئ البرمجة باستخدام C#

د. م. علي ذياب

محتويات المقرر

- نظرة عامة
- الخوارزميات
- مقدمة للغة البرمجة C#
- الأرقام و المتحولات في C#
- If statement
- Loops

نظرة عامة

محتويات المحاضرة

- لمحة عن بنية الحاسب و مكوناته
- نظم التشغيل (Operating System)
- لغات البرمجة
- المترجمات (Compilers)
- المفسرات (Interpreters)

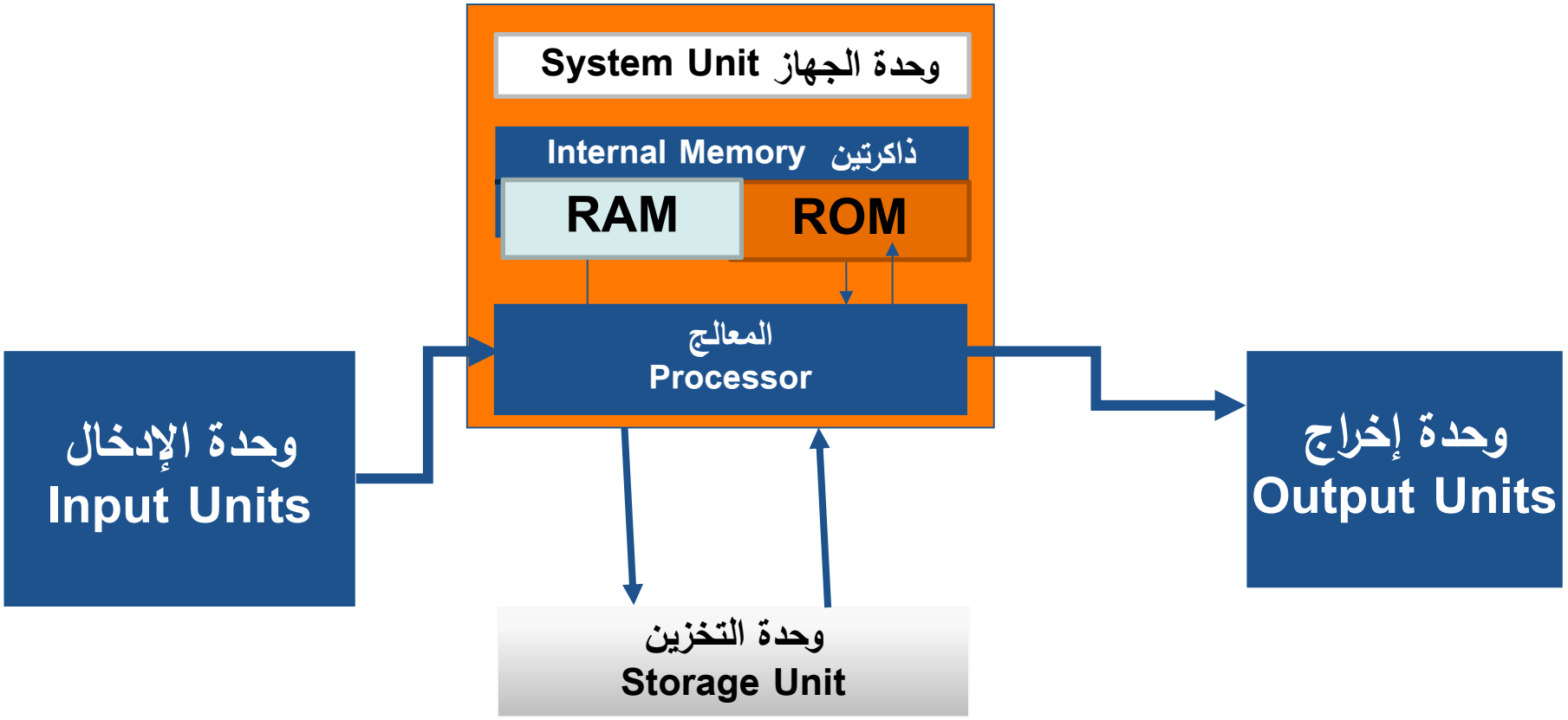
لمحة عن بنية الحاسب و مكوناته

الحاسب الآلي

- هو جهاز إلكتروني يقوم **باستقبال البيانات وتخزينها** ، ومن ثم إجراء مجموعة من العمليات الحسابية والمنطقية عليها وفقاً لسلسلة من التعليمات (البرامج) المختزنة في ذاكرته، ومن ثم يقوم بإخراج نتائج المعالجة على وحدات الإخراج المختلفة

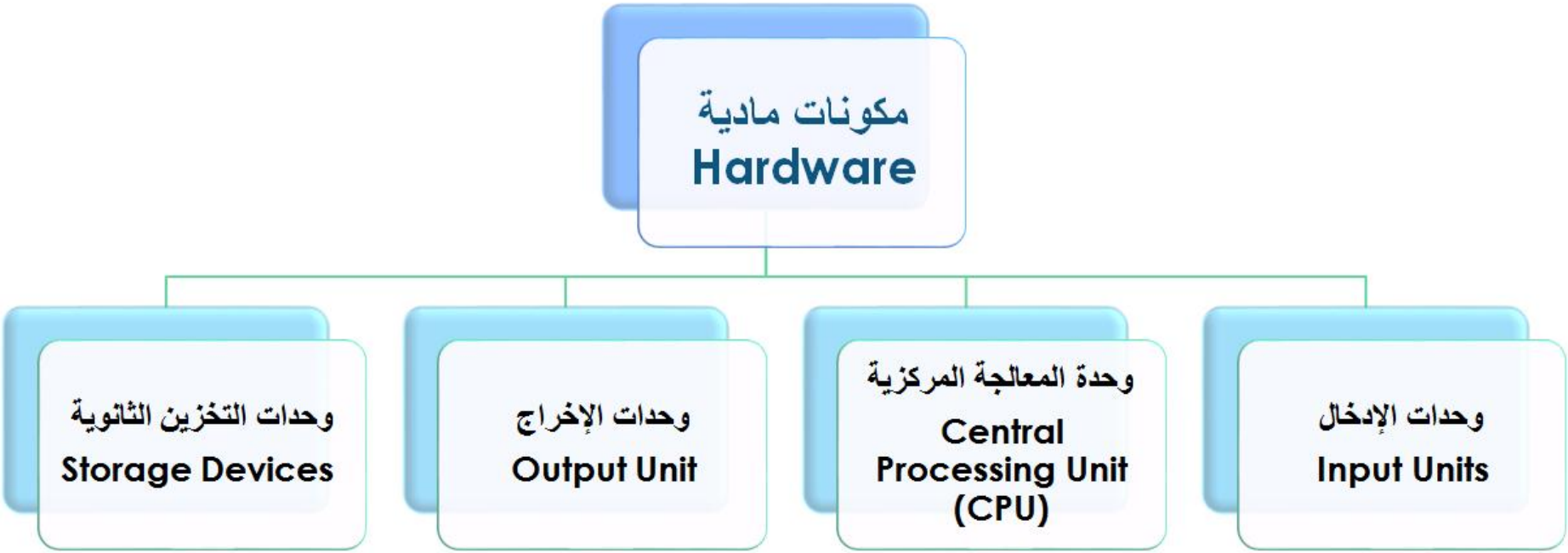
- **<u>العمليات الرئيسية التي يقوم بها الحاسب</u>**

العمليات الرئيسية التي يقوم بها الحاسب

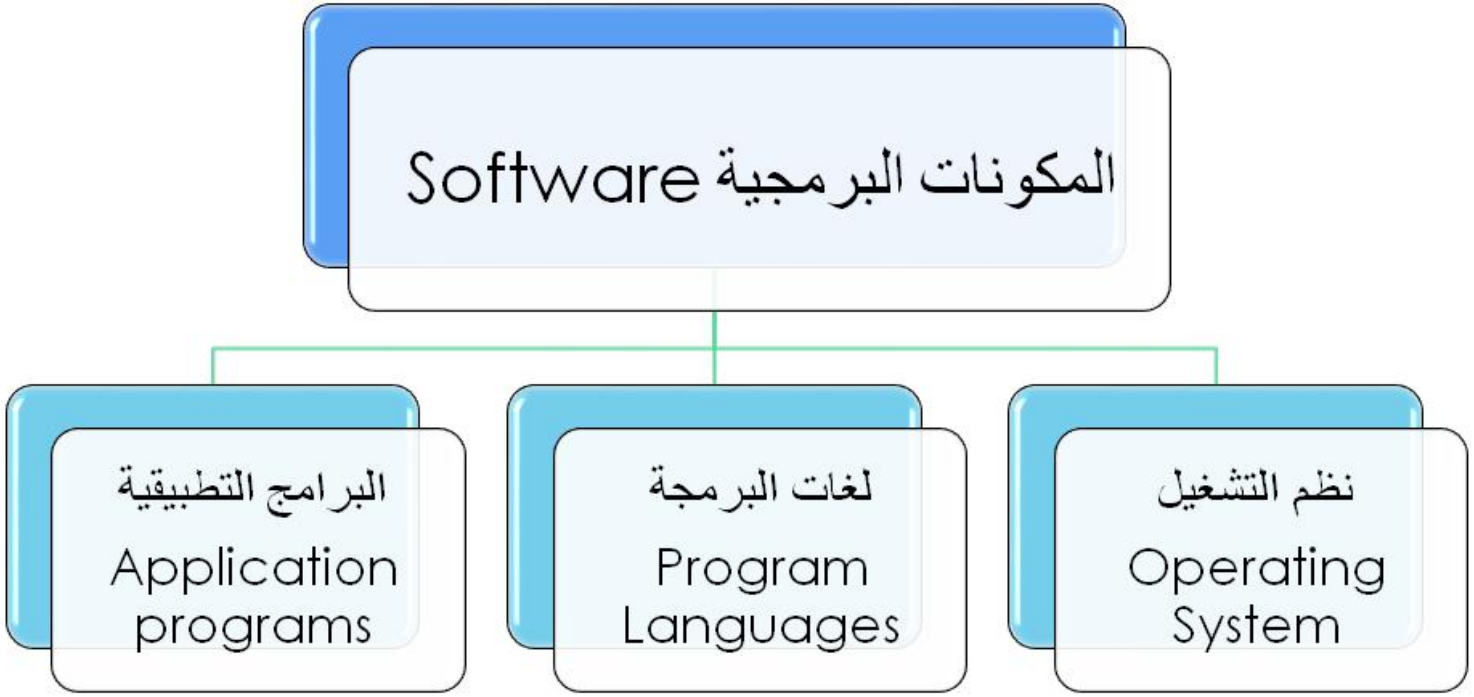


المكونات الرئيسية للحاسب الآلي

المكونات الرئيسية للحاسب الآلي



المكونات الرئيسية للحاسب الآلي



قياس بيانات الحاسب الآلي

- الوحدة الأساسية للقياس سعة الذاكره هي Bit وأساسها ثنائي، أي 0, 1
  - 1Byte = 8 Bits.
  - 1Kilo Byte (KB) = 1024 Byte.
  - 1Mega Byte (MB) = 1024 KB.
  - 1Giga Byte (GB) = 1024 MB.

- وحدات قياس سعة الذاكرة العشوائية RAM
- وحدة قياس سرعة CPU وهي الميجاهرتز MHz

نظم التشغيل (Operating System)

ماهي نظم التشغيل؟

- عبارة عن برنامج أو برامج متعددة قد تكون مخزنة على الحاسب الآلي ومسجلة على شريحة من نوع (ذاكرة قراءة فقط) وقد تكون محفوظة على القرص الصلب.

- التحكم بسير البيانات والأوامر بين البرامج التطبيقية وأجزاء الحاسب الآلي.

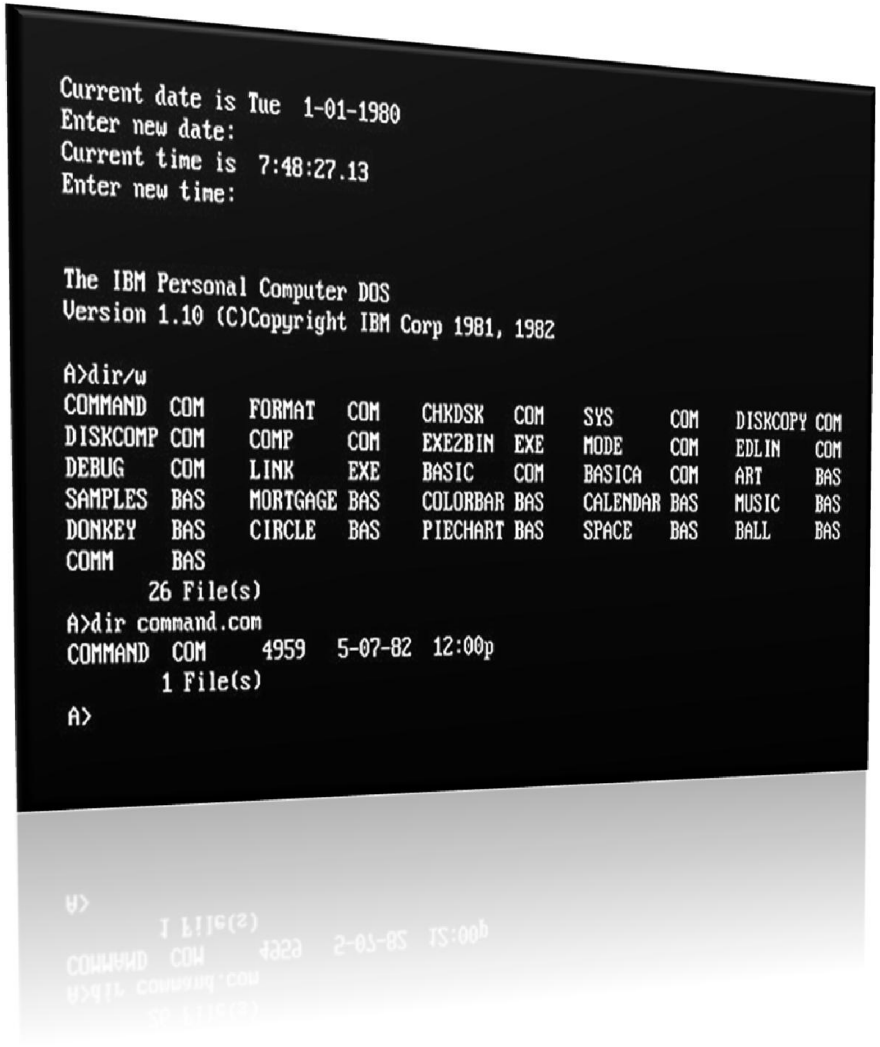
- وسيط بين المستخدم والحاسب الآلي.

وظائف نظم التشغيل

- استدعاء البرامج المراد تنفيذها من وحدة التخزين إلى الذاكرة الرئيسية ووضعها موضع التنفيذ
- مراقبة تنفيذ وظائف الإدخال والإخراج للبرامج المتعددة أثناء تنفيذها
- نقل الرسائل المتبادلة بين المشغل والبرامج المنفذة وبين بعضها
- المحافظة لكل برنامج على حقه في استخدام الوحدات والمساحة من الذاكرة المخصصة له
- التحكم في عملية التخزين والنسخ على الأقراص الممغنطة وترجمة أوامر التشغيل والبرامج

أنظمة التشغيل المشهورة

- نظام التشغيل (MS-DOS)



أنظمة التشغيل المشهورة

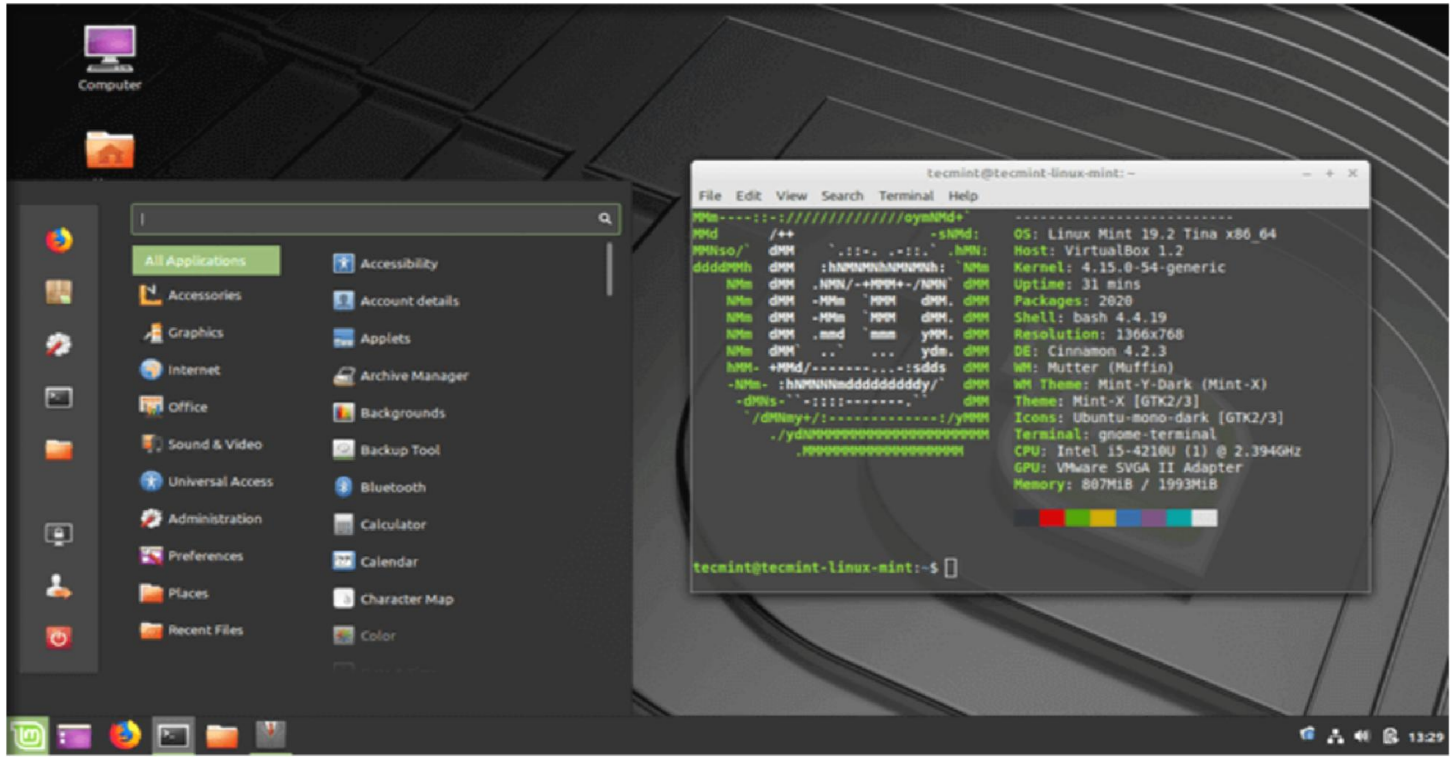
- نظام التشغيل (Windows)

أنظمة التشغيل المشهورة

- نظام التشغيل (Macintosh)

أنظمة التشغيل المشهورة

- نظام التشغيل (Linux)

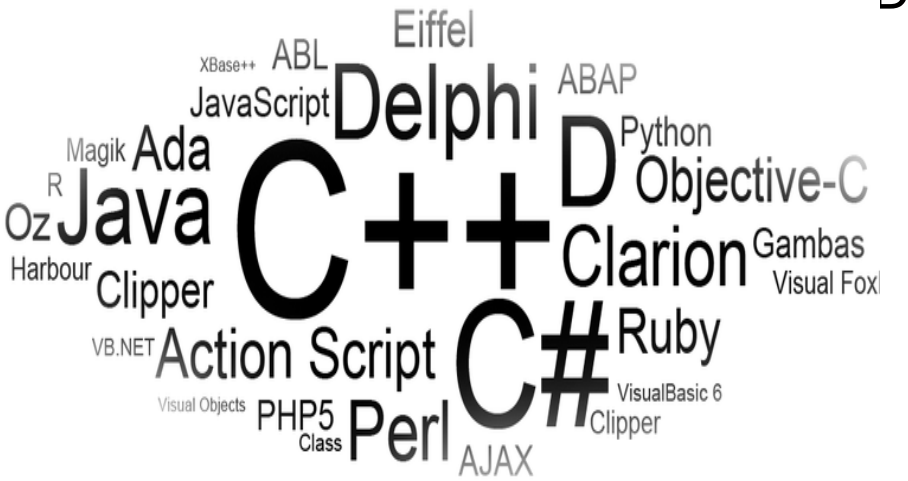


لغات البرمجة

لغات البرمجة

- وسيلة اتصال بين الحاسب والانسان

- أمثلة
  - C++
  - Java
  - Ruby
  - Delphi
  - ...



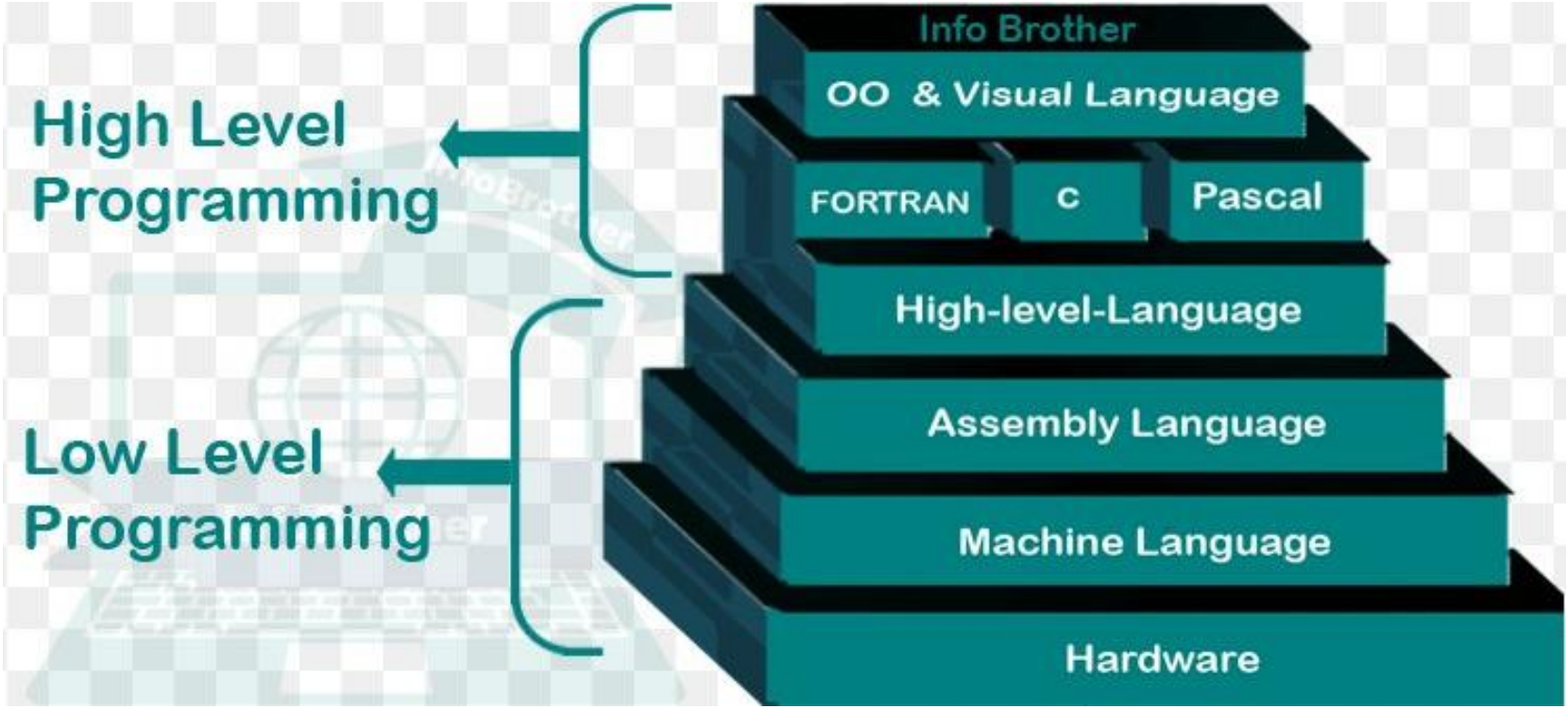
لغات البرمجة

- لغة البرمجة هي عبارة عن مجموعة من التعليمات و القواعد المستخدمة لتنفيذ جزء برمجي موجه لتنفيذ عملية ما
  - Programming language is a set of instructions and rules used (to implement blocks of software to perform certain operation)

- للغات البرمجية نوعين أساسيين
  - **لغات برمجة متدنية المستوى**
    - تتضمن لغتين, لغة الآلة (Machine language) ولغة التجميع (Assemply language)
    - لغة الآلة: هي اللغة التي تفهمها الـ CPU وتنفذها
    - لغة التجميع: المستوى الأول لترميز لغة الآلة, والذي يحتوي على تعليمات قابلة للقرائة (first level of coding of machine language into readable instructions)

لغات البرمجة

- **لغات برمجة عالية المستوى**
  - تُكتب بلغات قريبة إلى اللغات المستخدمة في الطبيعة
  - أبسط وأسهل للقراءة من لغات الآلة
  - تحتاج ترجمة (compile) وبناء (Build)



لغة الآلة

- اللغة التي تفهمها الـ CPU وتنفذها
- Set of “1“s and “0“s
- مثال
  - **10100001 00000000 00000000** (fetch the content of the address “0” and put them in the register AX)
  - **00000101 00000100 00000000** (add 4 to AX)
  - **10100011 00000000 00000000** (save the content of AX in the memory under the address “0”)
- كتابة برامج بلغة الآلة مهمة صعبة للغاية

لغة التجميع

- المستوى الأول للترميز بلغة الآلة

- تُكتب باستخدام تعليمات
  - **Commands**: MOV, SUB, XCHNG, etc.
  - **Register** names: AX, BX, CX, etc.
  - **Memory addresses**: [1000H], [2345H], etc.
  - **Data**: A DW 2 (define a variable with the name “A“ and the value “2“)

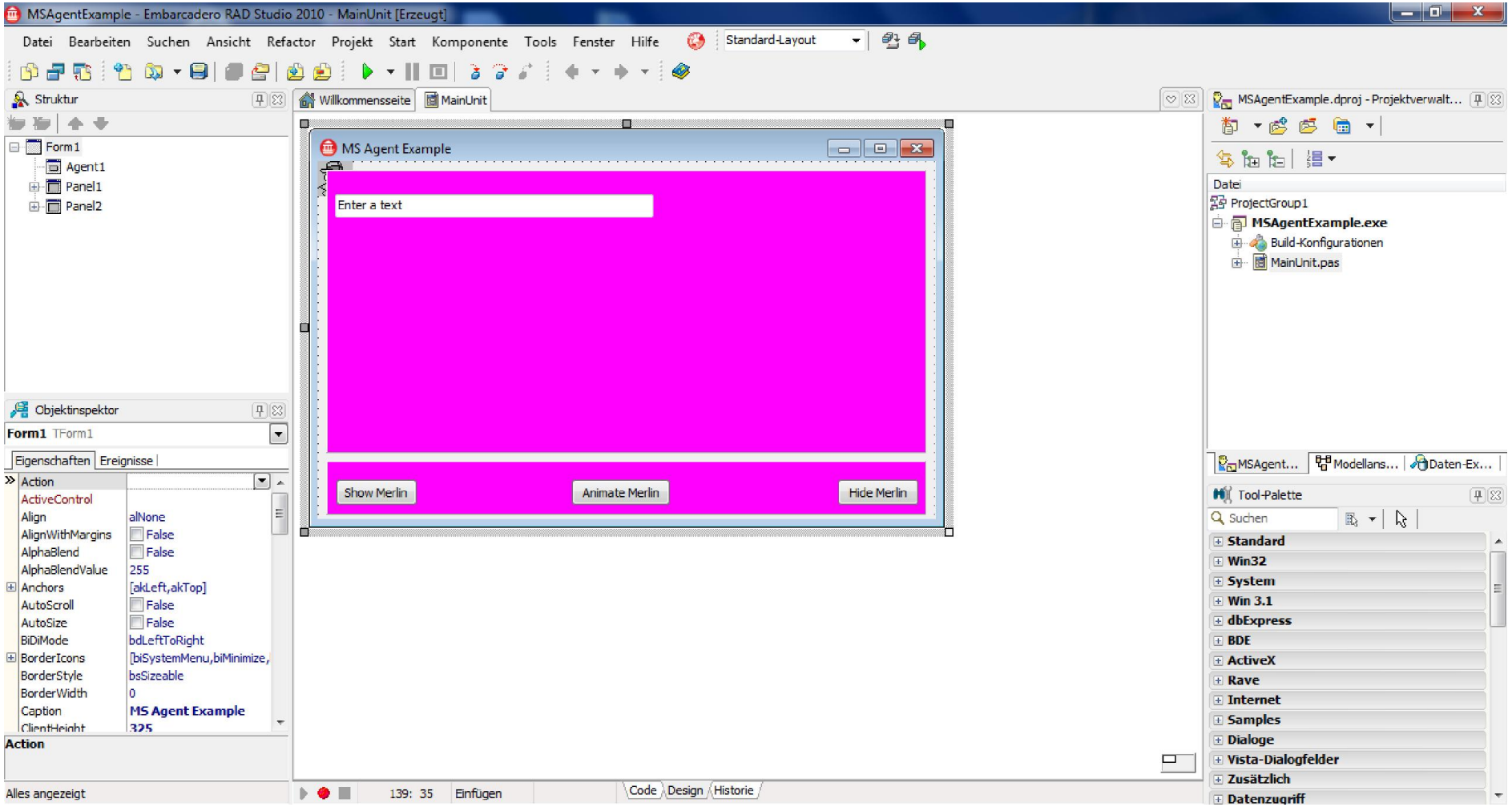
- البرامج المكتوبة بلغة التجميع أسرع من نظيراتها المكتوبة بلغة عالية المستوى

- البرامج المكتوبة بلغة التجميع يجب أن يتم تحويلها إلى برامج مكتوبة بلغة الآلة
  - باستخدام **Assempler**

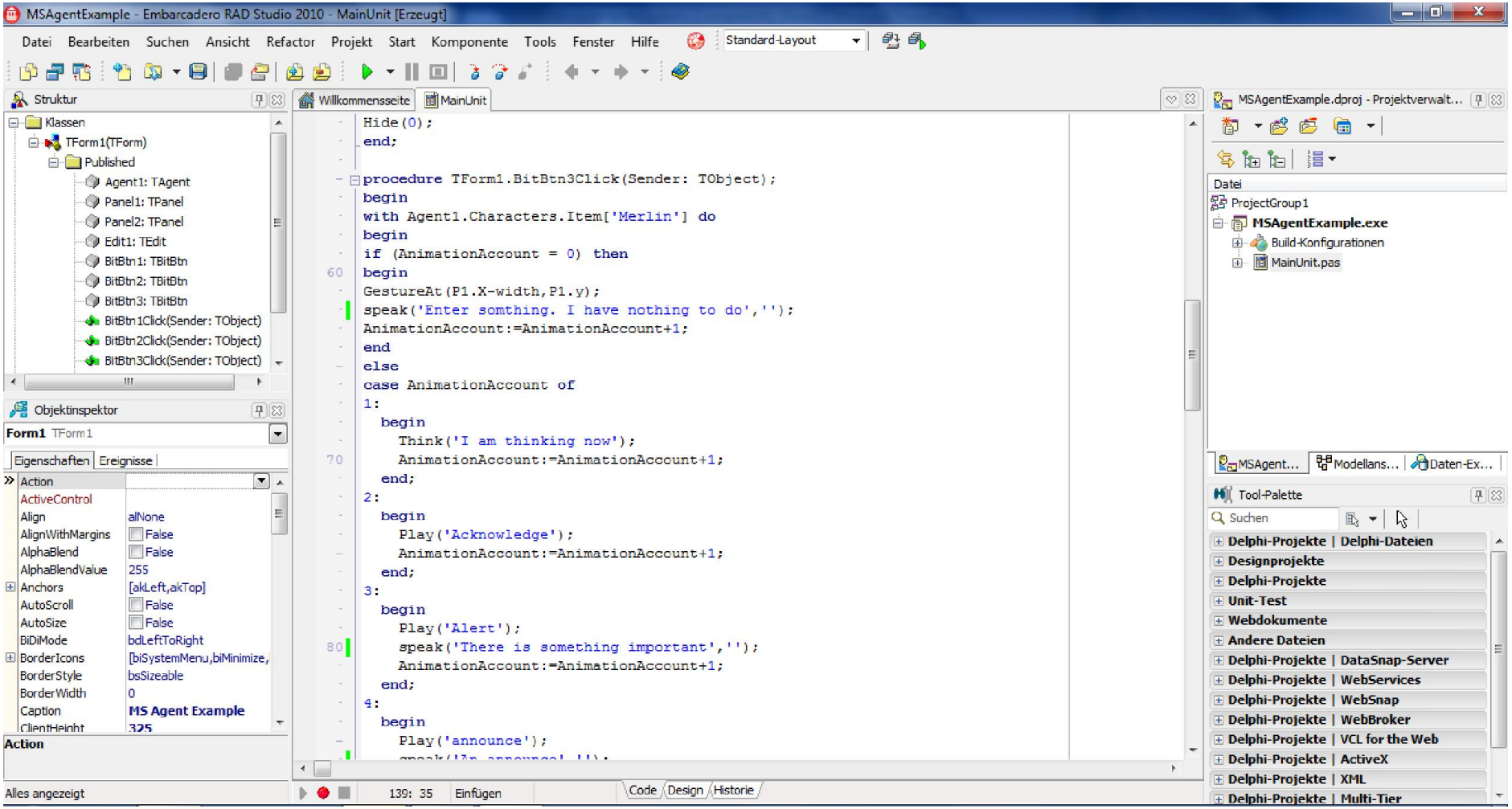
لغات البرمجة عالية المستوى

- تُكتب بلغات قريبة إلى اللغات المستخدمة في الطبيعة
- مستخدمة ومقبولة بشكل واسع
- أمثلة
  - C++, Delphi, Java, C#, PHP, TCL, etc.
- البرامج المكتوبة بلغة عالية المستوى تحتاج
  - ترجمة (compile) للتحقق من عدم وجود أخطاء (syntax errors)
  - وبناء (Build) ليتم تحويلها إلى برامج مكتوبة بلغة التجميع ثم إلى برامج مكتوبة بلغة الآلة

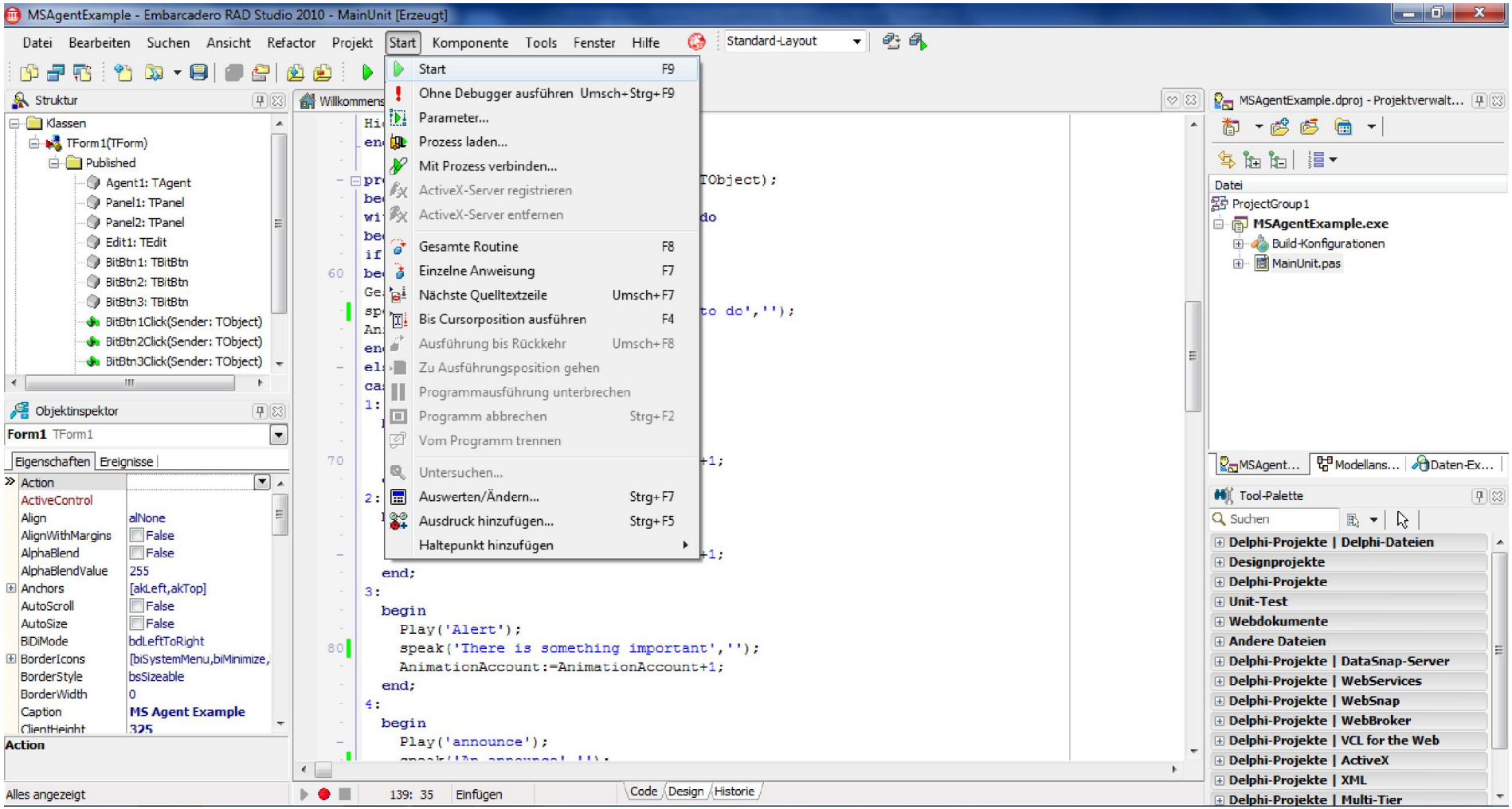
لغات البرمجة عالية المستوى



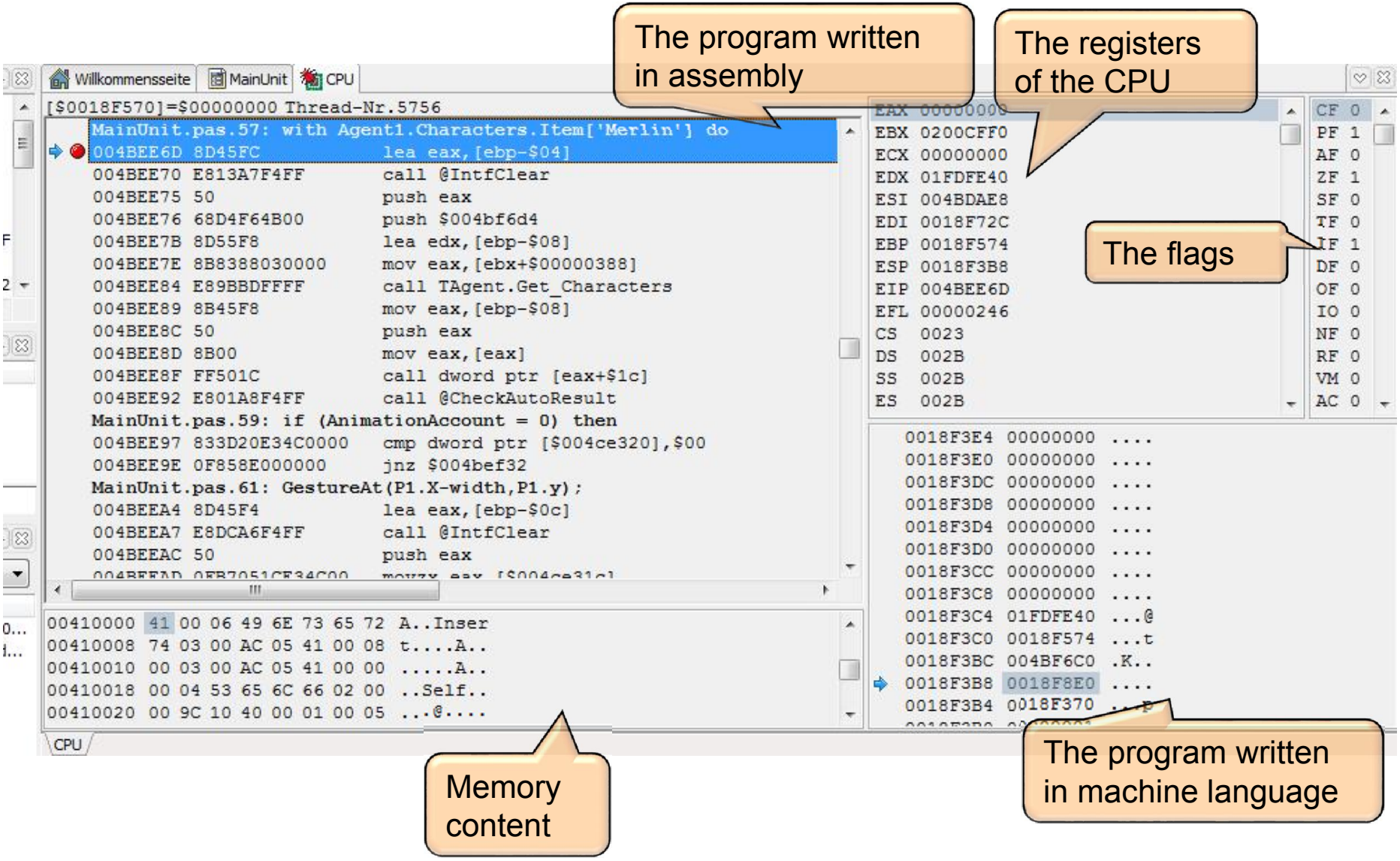
لغات البرمجة عالية المستوى



لغات البرمجة عالية المستوى



لغات البرمجة عالية المستوى



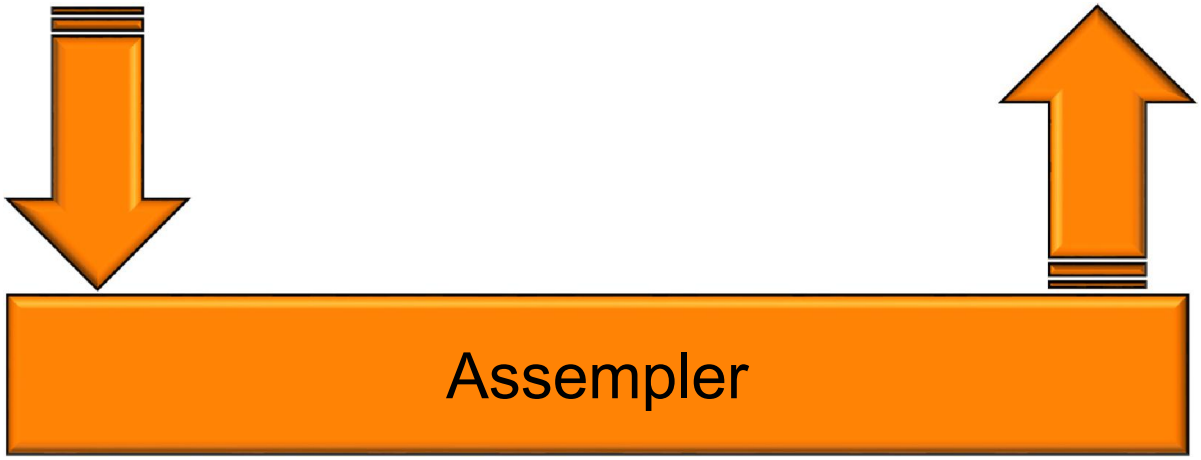
لغات البرمجة عالية المستوى

Programms written using assemply

```
MOV     Ax, [0000H]
ADD     Ax, 4
MOV     [0000H], AX
```

Programms written using assemply

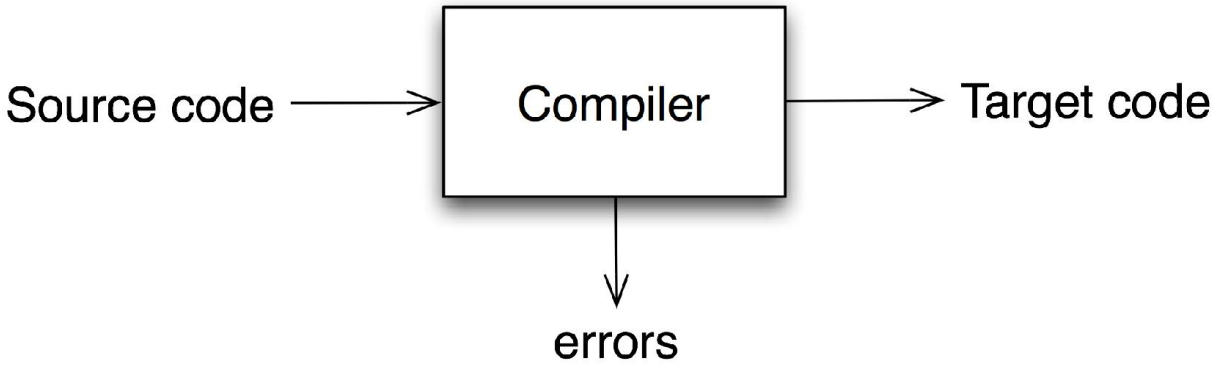
```
10100001 00000000 00000000
00000101 00000100 00000000
10100011 00000000 00000000
```



المترجمات (Compilers)

ماهو المترجم (Compiler)؟

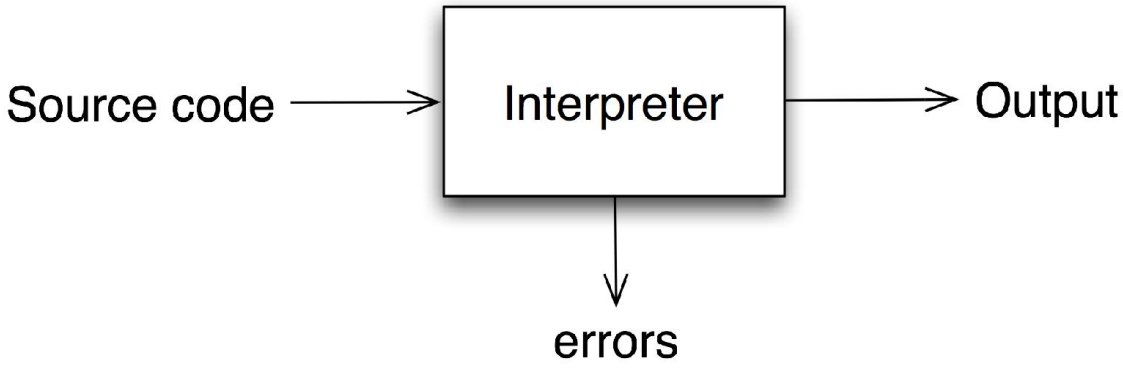
- عبارة عن برنامج يقوم **بترجمة** برنامج قابل للتنفيذ (executable program) مكتوب بلغة ما إلى برنامج قابل للتنفيذ مكتوب بلغة أخرى



المفسرات (Interpreters)

ماهو المفسر (Interpreter)؟

- عبارة عن برنامج يقوم **بقراءة** برنامج قابل للتنفيذ (executable program) مكتوب بلغة ما و **ينتج** نتائج تنفيذ هذا البرنامج



الخوارزميات

محتويات المحاضرة

- مفهوم الخوارزمية
- طرق صياغة الخوارزمية
- أمثلة

مفهوم الخوارزمية

ماهي الخوارزمية؟

- **الخوارزمية**: هي مجموعة من الخطوات يتم تنفيذها لحل مسألة معينة

- مراحل حل مسألة ما عن طريق الحاسوب

تعريف المسألة	أي تحديد مدخلات المسألة ومخرجاتها
التحليل	تحديد العمليات التي تؤدي الى حل المسألة
البرمجة	كتابة البرنامج بأي لغة برمجية
التنفيذ	أي البدء بإدخال المدخلات للبرنامج لحل المسألة
التوثيق	تكتب اهم المعلومات عن الخطوات السابقة بهدف الرجو اليها

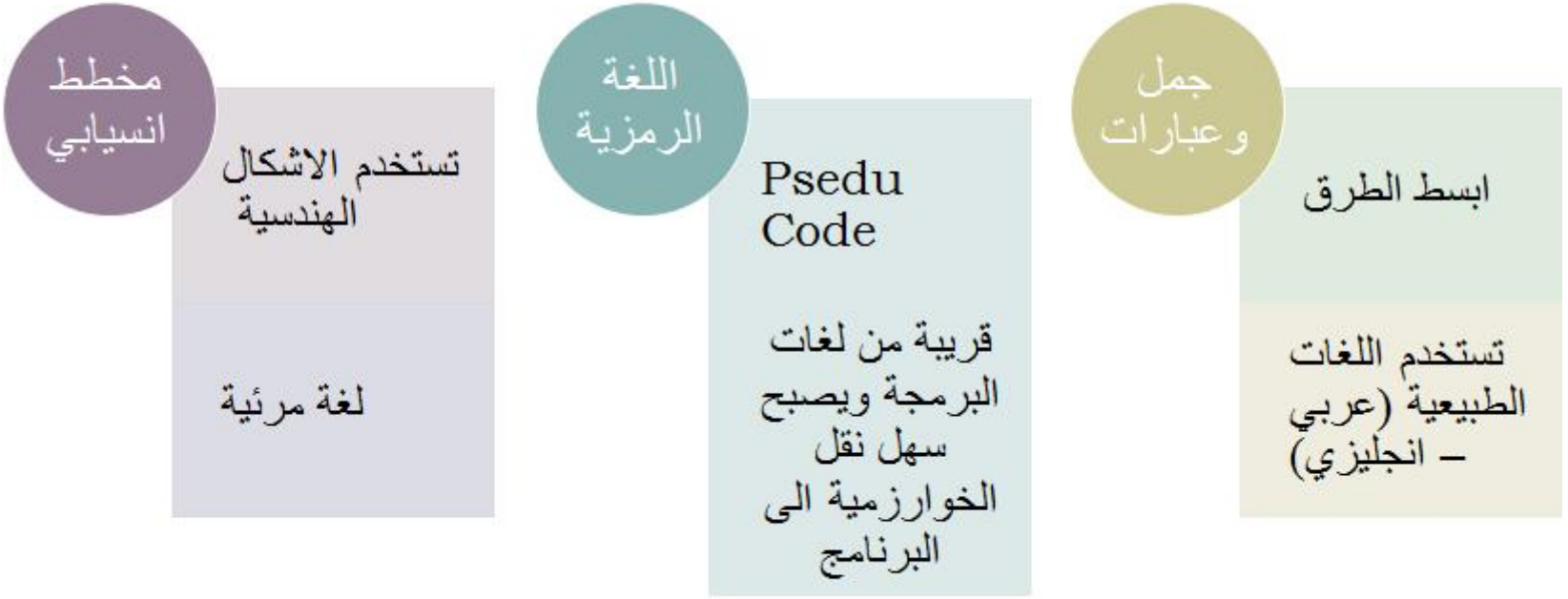
مثال (مسألة حسابية)

- **نريد ايجاد قيمة Z بمعلومية X و Y** **{ $Z=(X-Y)^2$ }**

1. تحديد المدخلات وهم X و Y
2. تحديد المخرجات Z
3. تحديد طريقة الحل (يمكن أن يكون هناك أكثر من طريقة)
   - تعريض قيمة كل من X و Y
   - ايجاد ناتج (X-Y)
   - ايجاد ناتج تربيع الخطوة السابقة وهي قيمة Z

طرق صياغة الخوارزمية

طرق إنشاء الخوارزمية



استخدام اللغة الطبيعية

- طريقة مباشرة للتعبير عن الخوارزمية بجمل وعبارات قصيرة وواضحة ومفهومة.

- **مثال (خوارزمية اعداد رسالة الكترونية)**

1. **البداية**
2. **كتابة الرسالة**
3. **كتابة عنوان الرسالة**
4. **كتابة عنوان مستلم الرسالة**
5. **ارسال الرسالة**
6. **النهاية**

ملاحظات على الخوارزمية

- كل خطوة فيها عملية وحدة فقط
- ترتيب الخطوات بشكل منطقي
- من الممكن تفصيل بعض الخطوات مثلا (خطوة كتابة الرسالة تفصيلها الى كتابة التحية ونص الرسالة وشكر في الاخير)

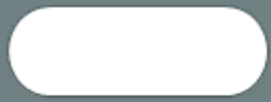
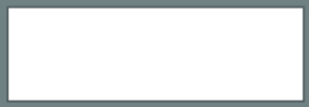
مزايا وعيوب اللغة الطبيعية للخوارزميات

صعوبة الوصف بدقة ووضوح احيانا يكون فيها بعض الغموض	طريقة سهله في متناول الجميع ولا تحتاج الى خبرة

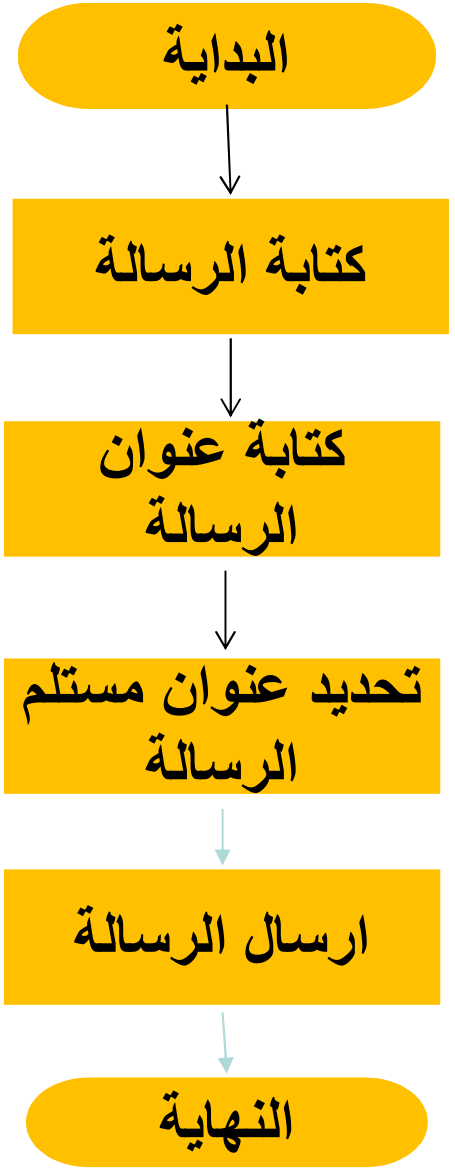
المخطط الانسيابي (Flowchart)

- تستخدم فيه الاشكال الهندسية (مستطيل – مربع – معين ..) للتعبير عن الخوارزمية وحل مسألة معينة

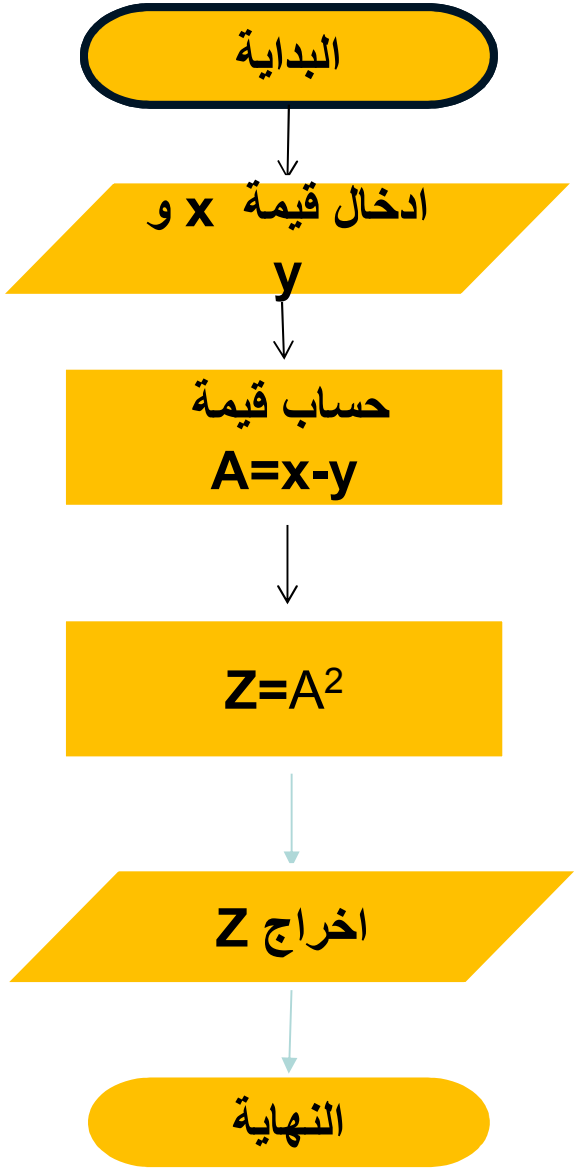
- له عدة فوائد
  - تنظيم سير الحل لأي مسألة
  - وتسهيل صياغة الخوارزمية

				
يستخدم للتعبير عن بداية الخوارزمية ونهايتها	للدلالة على اجراء عملية (معالجة حسابية)	للدلالة على عملية ادخال أو اخراج بيانات	لاختبار شرط أو عملية مقارنة ولأي عملية فيها اتخاذ قرار	عند وصل مخطط انسيابي مع مخطط اخر (موصل)

مثال (كتابة رسالة الكترونية)



ايجاد قيمة $Z=(X-Y)^2$



لغة رمزية (Pseudocode)

- لغة رمزية ليسس لها مترجم و تستخدم بعض العبارات القريبة من اللغات الطبيعية

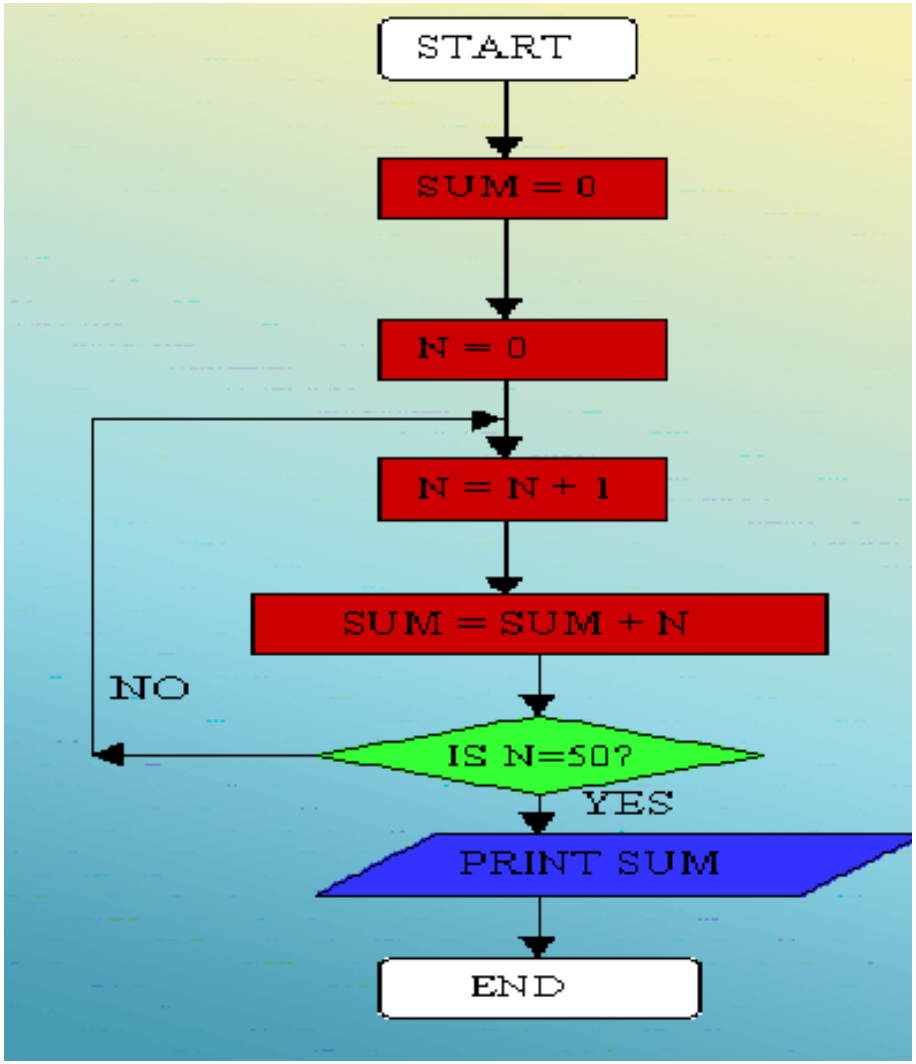
More Than 10

An algorithm that detects if the value inputted is greater than 10

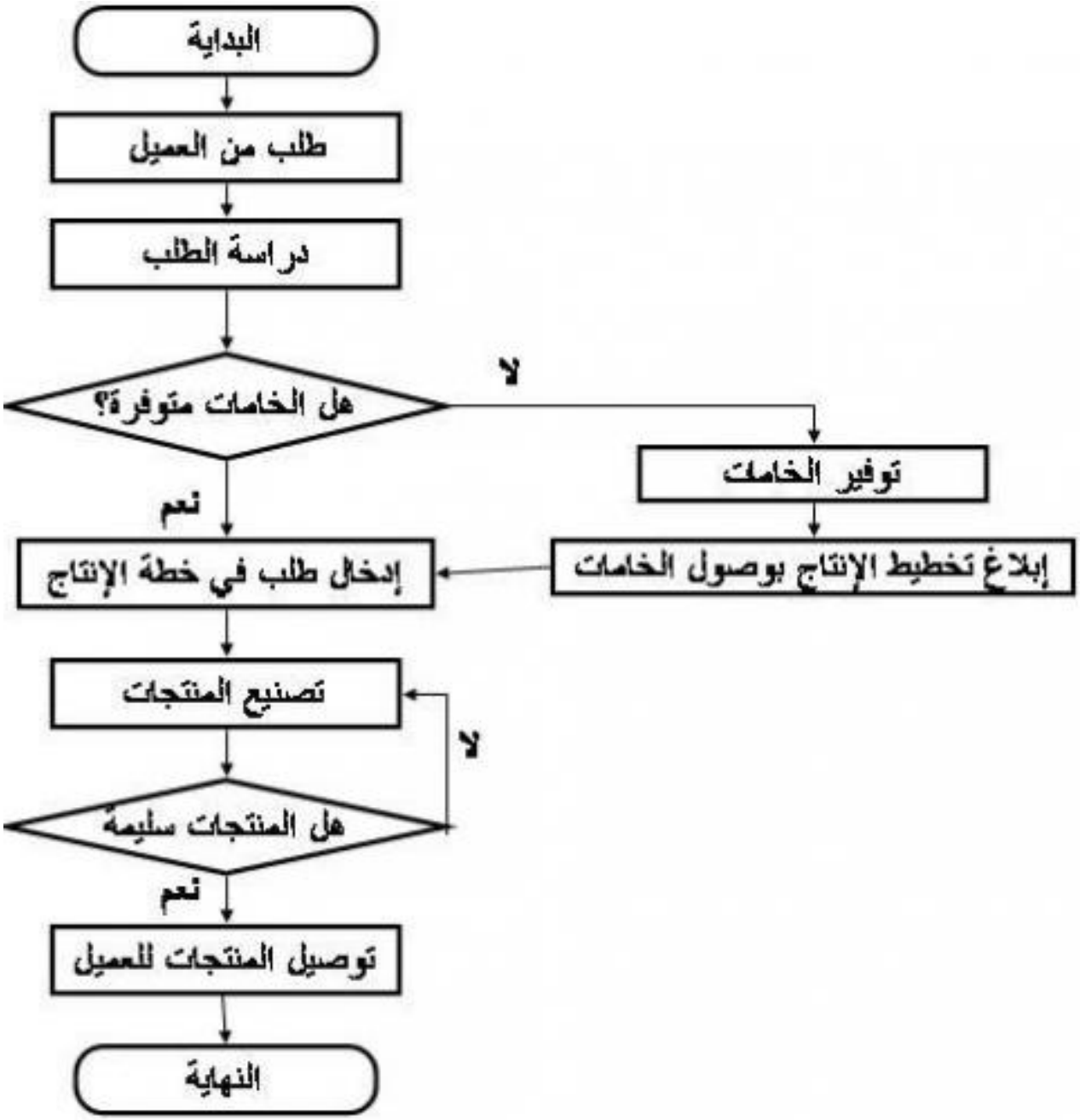
```
INPUT number
IF number > 10 THEN
   OUTPUT "Yes"
ELSE
   OUTPUT "No"
```

أمثلة

مثال 1: جمع الأعداد من 1 الى 50



مثالل 2


البداية
طلب من العميل
دراسة الطلب
هل الخامات متوفرة؟
لا
نعم
توفير الخامات
إبلاغ تخطيط الإنتاج بوصول الخامات
إدخال طلب في خطة الإنتاج
تصنيع المنتجات
لا
هل المنتجات سليمة
نعم
توصيل المنتجات للعميل
النهاية

مقدمـــــة للغة البرمجة C#

محتويات المحاضرة

- حول لغة البرمجة C#
- برنامج Hello World

حول لغة البرمجة C#

حول لغة البرمجة C#

- لغة تعبيرية سهلة و قوية
  - سهلة التعلم لمن لديه معرفة بـ C, C++, Java or JavaScript

- لغة غرضية التوجه

- محتويات البرمجية
  - التعليمات البرمجية
  - بنية البيانات
  - التوثيق

برنامج Hello World

برنامج Hello World

```
using System;

class Hello
{
    static void Main()
    {
        Console.WriteLine("Hello, World");
    }
}
```

برنامج Hello World

```
using System;

class Hello
{
    static void Mai
    {
        Console.WriteLine("Hello, World");
    }
}
```

```
Hello, World
```

برنامج Hello World

Using directive that references the **System** namespace

```
using System;

class Hello
{
    static void Main()
    {
        Console.WriteLine("Hello, World");
    }
}
```

namespace

- تزود بنية هرمية لتنظيم برامج و مكتبات C#

- **System** namespace
  - تحتوي عدة أنماط
    - Consloe
    - IO
    - Collections

- استخدام **Using** directive يمكننا من كتابة
  - Console.WriteLine("Hello, World")
  - عوضاً عن
  - System.Console.WriteLine("Hello, World")

برنامج Hello World

Class **Hello**

يحتوي عنصر وحيد هو الـ Methode المسماة Main

```
class Hello
{
    static void Main()
    {
        Console.WriteLine("Hello, World");
    }
}
```

برنامج Hello World

Static modifier

Void means no output returned

```
        static void Main()
        {
            Console.WriteLine("Hello, World");
        }
}
```

برنامج Hello World

```
using System;

class Hello
{
    static void Main()
    {
        Console.WriteLine("Hello, World");
    }
}
```

The methode **Writeline** of **Console** class in the **system** namespace

الأرقام في C#

محتويات المحاضرة

- أنماط المعطيات في لغة البرمجة C#
- الأعداد الصحيحة
- الأعداد الحقيقة
- النمط المرجعي (Reference Type)

أنماط المعطيات في لغة البرمجة C#

أنماط المعطيات

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0}$ to 28	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F

أنماط المعطيات

int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

الأعداد الصحيحة

الأعداد الصحيحة

```
int a = 18;
int b = 6;
int c = a + b;
Console.WriteLine(c);
```

الأعداد الصحيحة

تعريف متحولين صحيحين (a, b), و تعريف متحول ثالث (c) يُعطي نتيجة الجمع

```
int a = 18;
int b = 6;
int c = a + b;
Console.WriteLine(c);
```

تعريف متحولين صحيحين (a, b), و تعريف متحول ثالث (c) يُعطي نتيجة الجمع

```
int a = 18;
int b = 6;
int c = a + b;
Console.WriteLi
```

```
24
```

الأعداد الصحيحة

```
int a = 18;
int b = 6;
int c = a + b;
Console.WriteLine(
```

- (-) for subtraction
- (*) for multiplication
- (/) for division
- (%) for mod

ترتيب الأولويات

```
int a = 5;
int b = 4;
int c = 2;
int d = a + b * c;
Console.WriteLine(d);
```

ترتيب الأولويات

```
int a = 5;
int b = 4;
int c = 2;
int d = a + b * c;
Console.WriteLi
```

ترتيب الأولويات

```
int a = 5;
int b = 4;
int c = 2;
int d = (a + b) * c;
Console.WriteLine(d);
```

ترتيب الأولويات

```
int a = 5;
int b = 4;
int c = 2;
int d = (a + b) * c;
Console.WriteLine(d);
```

18

ترتيب الأولويات

```
int d = (a + b) - 6 * c + (12 * 4) / 3 + 12;
```

25

ترتيب الأولويات

```
int a = 7;
int b = 4;
int c = 3;
int d = (a + b) / c;
Console.WriteLine(d);
```

ترتيب الأولويات

```
int a = 7;
int b = 4;
int c = 3;
int d = (a + b) / c;
Console.WriteLine
```

```
3
```

ترتيب الأولويات

```
1  using System;
2
3
4      class Boxtester {
5
6          static void Main() {
7  int a = 7;
8  int b = 4;
9  int c = 3;
10 int d = (a + b) % c;
11 Console.WriteLine(d);
12         }
13     }
```

ترتيب الأولويات

```
1  using System;
2
3
4      class Boxtester {
5
6          static void Main() {
7  int a = 7;
8  int b = 4;
9  int c = 3;
10 int d = (a + b)
11 Console.WriteLi
12         }
13     }
```

```
2
```

دقة الأرقام الصحيحة و حدودها

```
int max = int.MaxValue;
int min = int.MinValue;
Console.WriteLine($"The range of integers is {min} to {max}");
```

دقة الأرقام الصحيحة و حدودها

```
int max = int.MaxValue;
int min = int.MinValue;
Console.WriteLine($"The range of integers is {min} to {max}");
```

The range of integers is **-2,147,483,648 to 2,147,483,647**

تابع sizeof

```
using System;

namespace DataTypeApplication {
   class Program {
      static void Main(string[] args) {
         Console.WriteLine("Size of int: {0}", sizeof(int));
         Console.ReadLine();
      }
   }
}
```

تابع sizeof

```
using System;

namespace DataTypeApplication {
   class Program {
      static void Main(string[] args) {
         Console.WriteLine("Size of int: {0}", sizeof(int));
         Console.ReadLine();
      }
   }
}
```

```
Size of int: 4
```

الأعداد الحقيقية

التعامل مع الأعداد الحقيقية

```
double a = 5;
double b = 4;
double c = 2;
double d = (a + b) / c;
Console.WriteLine(d);
```

4.5

التعامل مع الأعداد الحقيقية

```
double a = 19;
double b = 23;
double c = 8;
double d = (a + b) / c;
Console.WriteLine(d);
```

```
5.25
```

حدود الأعداد الحقيقية

```
double max = double.MaxValue;
double min = double.MinValue;
Console.WriteLine($"The range of double is {min} to {max}");
```

- 5.0 E-324 to 1.7 E 308

If Statement

معاملات المقارنة

Operator	Action
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

مثال

```
int weight = 700;
Console.WriteLine(weight >= 500); // True

char gender = 'm';
Console.WriteLine(gender <= 'f'); // False

double colorWaveLength = 1.630;
Console.WriteLine(colorWaveLength > 1.621); // True

int a = 5;
int b = 7;
bool condition = (b > a) && (a + b < a * b);
Console.WriteLine(condition); // True

Console.WriteLine('B' == 'A' + 1); // True
```

مثال

```
int weight = 700;
Console.WriteLine(weight >= 500); // True

char gender = 'm';
Console.WriteLine(gender <= 'f'); // False

d
C

i
i
b
C

Console.WriteLine('B' == 'A' + 1); // True
```

```
True
Fasle
True
True
True
```

مثال (مقارنة بين أعداد صحيحة و محارف)

```
Console.WriteLine("char 'a' == 'a'? " + ('a' == 'a'));  // True
Console.WriteLine("char 'a' == 'b'? " + ('a' == 'b'));  // False
Console.WriteLine("5 != 6? " + (5 != 6));               // True
Console.WriteLine("5.0 == 5L? " + (5.0 == 5L));         // True
Console.WriteLine("true == false? " + (true == false)); // False
```

مثال (مقارنة بين أعداد صحيحة و محارف)

```
Console.WriteLine("char 'a' == 'a'? " + ('a' == 'a'));  // True
Console.WriteLine("char 'a' == 'b'? " + ('a' == 'b'));  // False
Console.WriteLine("5 != 6? " + (5 != 6));               // True
Console.WriteLine("5.0 == 5L? " + (5.0 == 5L));         // True
Console.WriteLine("true == false? " + (true == false)); // False
```

```
char 'a' == 'a'? True
char 'a' == 'b'? False
5 != 6? True
5.0 == 5L? True
true == false? False
```

مثال (مقارنة بين References to Objects)

```
string str = "beer";
string anotherStr = str;
string thirdStr = "bee";
thirdStr = thirdStr + 'r';
Console.WriteLine("str = {0}", str);
Console.WriteLine("anotherStr = {0}", anotherStr);
Console.WriteLine("thirdStr = {0}", thirdStr);
Console.WriteLine(str == anotherStr); // True - same object
Console.WriteLine(str == thirdStr); // True - equal objects
Console.WriteLine((object)str == (object)anotherStr); // True
Console.WriteLine((object)str == (object)thirdStr); // False
```

مثال (مقارنة بين References to Objects)

```
string str = "beer";
string anotherStr = str;
string thirdStr = "bee";
thirdStr = thirdStr + 'r';
Console.WriteLine("str = {0}", str);
Console.WriteLine("anotherStr = {0}", anotherStr);
```

```
str = beer
anotherStr = beer
thirdStr = beer
True
True
True
False
```

مثال (المعاملات المنطقية)

- Logical operators && (logical AND) and || (logical OR) are only used on Boolean expressions

```
bool result = (2 < 3) && (3 < 4);
```

```
bool result = (2 < 3) || (1 == 2);
```

- Operators & (AND) and | (OR) are similar

مثال (المعاملات المنطقية)

- The ^ operator, known as exclusive OR (XOR)

```
Console.WriteLine("Exclusive OR: "+ ((2 < 3) ^ (4 > 3)));
```

Exclusive OR: False

- The operator ! returns the reversed value of the Boolean expression

```
bool value = !(7 == 5); // True
Console.WriteLine(value);
```

If & if-else

```
if (Boolean expression)
{
   Body of the conditional statement;
}
```

```
static void Main()
{
   Console.WriteLine("Enter two numbers.");
   Console.Write("Enter first number: ");
   int firstNumber = int.Parse(Console.ReadLine());
   Console.Write("Enter second number: ");
   int secondNumber = int.Parse(Console.ReadLine());
   int biggerNumber = firstNumber;
   if (secondNumber > firstNumber)
   {
      biggerNumber = secondNumber;
   }
   Console.WriteLine("The bigger number is: {0}", biggerNumber);
}
```

If & if-else

```
if (Boolean expression)
{
   Body of the conditional statement;
}
```

```
static void Main()
{
   Console.WriteLine("Enter two numbers.");
   Console.Write("Enter first number: ");
   ...
}
```

```
Enter two numbers.
Enter first number: 4
Enter second number: 5
The bigger number is: 5
```

If & if-else

```
int a = 6;
if (a > 5)
   Console.WriteLine("The variable is greater than 5.");
   Console.WriteLine("This code will always execute!");
// Bad practice: misleading code
```

The variable is greater than 5
This code will always execute!

If & if-else

```
if (Boolean expression)
{
   Body of the conditional statement;
}
else
{
   Body of the else statement;
}
```

If & if-else

```
static void Main()
{
   int x = 2;
   if (x > 3)
   {
      Console.WriteLine("x is greater than 3");
   }
   else
   {
      Console.WriteLine("x is not greater than 3");
   }
}
```

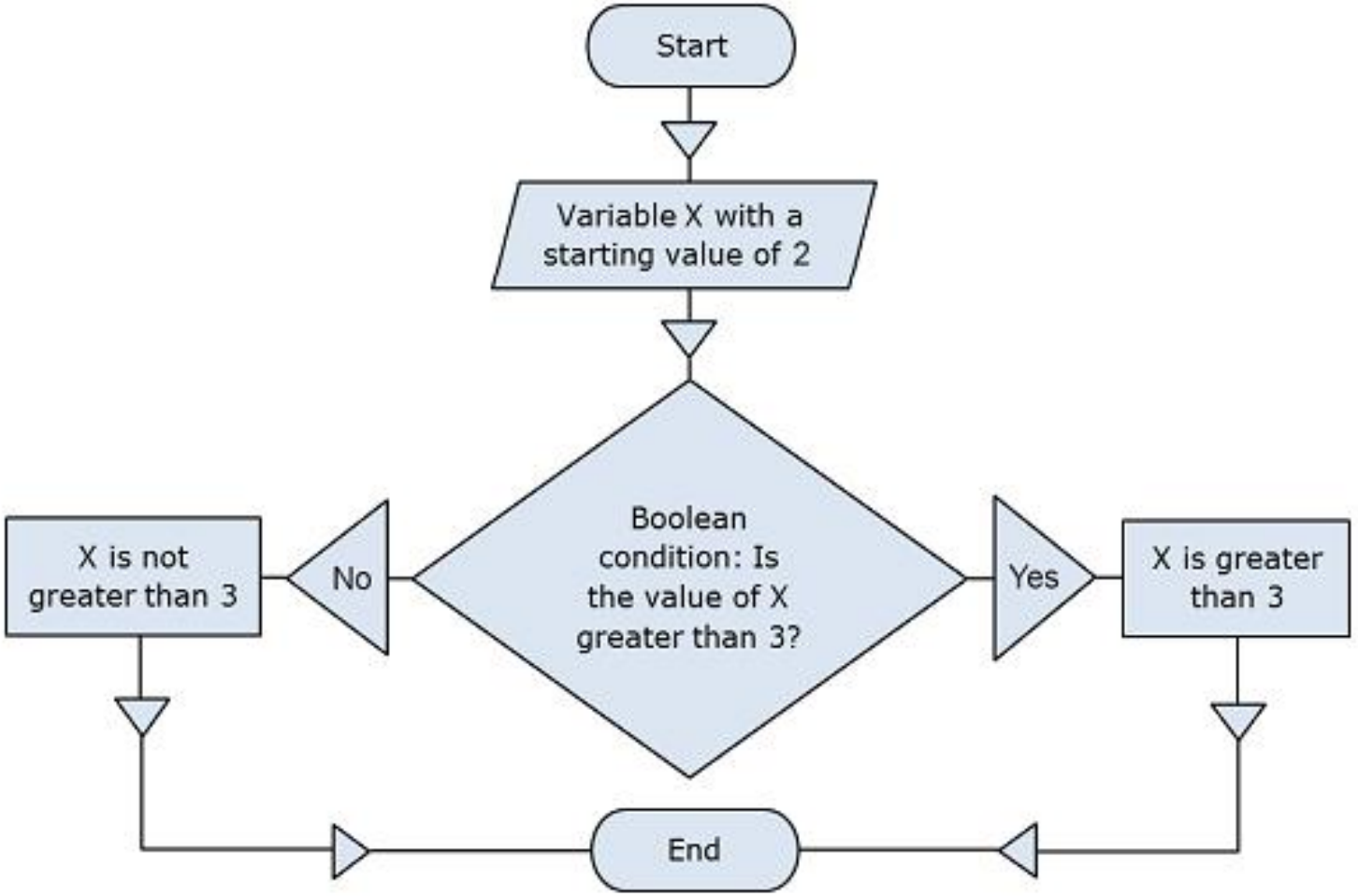
If & if-else

```
static void Main()
{
   int x = 2;
   if (x > 3)
   {
      Console.WriteLine("x is greater than 3");
   }

}
```

```
x is not greater than 3
```

If & if-else


Start
Variable X with a starting value of 2
Boolean condition: Is the value of X greater than 3?
No
X is not greater than 3
Yes
X is greater than 3
End

Nested if

```
int first = 5;
int second = 3;

if (first == second)
{
   Console.WriteLine("These two numbers are equal.");
}
else
{
   The first number is greater.
}
```

Nested if

```
char ch = 'X';
if (ch == 'A' || ch == 'a')
{
   Console.WriteLine("Vowel [ei]");
}
else if (ch == 'E' || ch == 'e')
{
   Console.WriteLine("Vowel [i:]");
}
else if (ch == 'I' || ch == 'i')
{
   Console.WriteLine("Vowel [ai]");
```

Constant

```
   Console.WriteLine("Consonant");
}
```

Switch-case

```
switch (integer_selector)
{
   case integer_value_1:
      statements;
      break;
   case integer_value_2:
      statements;
      break;
   // …
   default:
      statements;
      break;
}
```

Switch-case

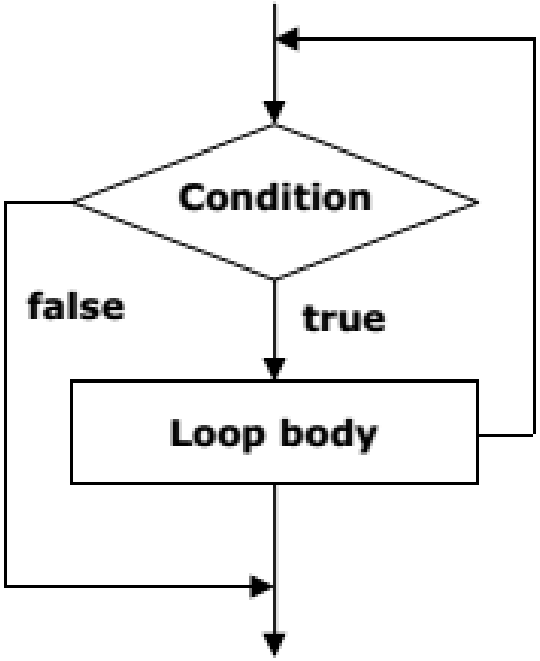
```
int number = 6;
switch (number)
{
   case 1:
   case 4:
   case 6:
   case 8:
   case 10:
```

The number is not prime!

loops

While loop

```
while (condition)
{
   loop body;
}
```



While loop

```
// Initialize the counter
int counter = 0;

// Execute the loop body while the loop condition holds
while (counter <= 9)
{
   // Print the counter value
   Console.WriteLine("Number : " + counter);
   // Increment the counter
   counter++;
}
```

While loop

```
// Initialize the counter
int counter = 0;

// Execute the loop body while the loop condition holds
```

```
Number : 0
Number : 1
Number : 2
Number : 3
Number : 4
Number : 5
Number : 6
Number : 7
Number : 8
Number : 9
```

While loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
int num = 1;
int sum = 1;
Console.Write("The sum 1");
while (num < n)
{
   num++;
   sum += num;
   Console.Write(" + " + num);
}
Console.WriteLine(" = " + sum);
```

While loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
int num = 1;
int sum = 1;
Console.Write("The sum 1");
while (num < n)
{
   num++;
```

N = 17
The sum 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 + 17 = 153

While loop

```
Console.Write("Enter a positive number: ");
int num = int.Parse(Console.ReadLine());
int divider = 2;
int maxDivider = (int)Math.Sqrt(num);
bool prime = true;
while (prime && (divider <= maxDivider))
{
   if (num % divider == 0)
   {
      prime = false;
   }
   divider++;
}
Console.WriteLine("Prime? " + prime);
```

While loop

```
Console.Write("Enter a positive number: ");
int num = int.Parse(Console.ReadLine());
int divider = 2;
int maxDivider = (int)Math.Sqrt(num);
bool prime = true;
while (prime && (divider <= maxDivider))
{
   if (num % divider == 0)
```

```
Enter a positive number: 37
Prime? True
Enter a positive number: 34
Prime? False
```

While loop

```
int n = int.Parse(Console.ReadLine());
// "decimal" is the biggest C# type that can hold integer values
decimal factorial = 1;
// Perform an "infinite loop"
while (true)
{
   if (n <= 1)
   {
      break;
   }
   factorial *= n;
   n--;
}
Console.WriteLine("n! = " + factorial);
```

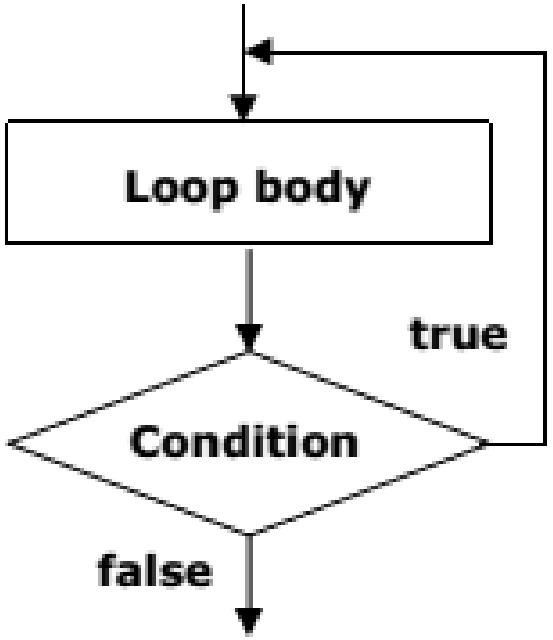
While loop

```
int n = int.Parse(Console.ReadLine());
// "decimal" is the biggest C# type that can hold integer values
decimal factorial = 1;
// Perform an "infinite loop"
while (true)
{
   if (n <= 1)
   {
```

```
10
n! = 3628800
```

Do-While loop

```
do
{
   executable code;
} while (condition);
```



Do-While loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
decimal factorial = 1;
do
{
   factorial *= n;
   n--;
} while (n > 0);
Console.WriteLine("n! = " + factorial);
```

Do-While loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
decimal factorial = 1;
do
{
   factorial *= n;
   n--;
} while (n > 0);
```

```
n = 7
n! = 5040
```

Do-While loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());

Console.Write("m = ");
int m = int.Parse(Console.ReadLine());

int num = n;
long product = 1;
do
{
   product *= num;
   num++;
} while (num <= m);

Console.WriteLine("product[n...m] = " + product);
```

Do-While loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());

Console.Write("m = ");
int m = int.Parse(Console.ReadLine());

int num = n;
long product = 1;
do
```

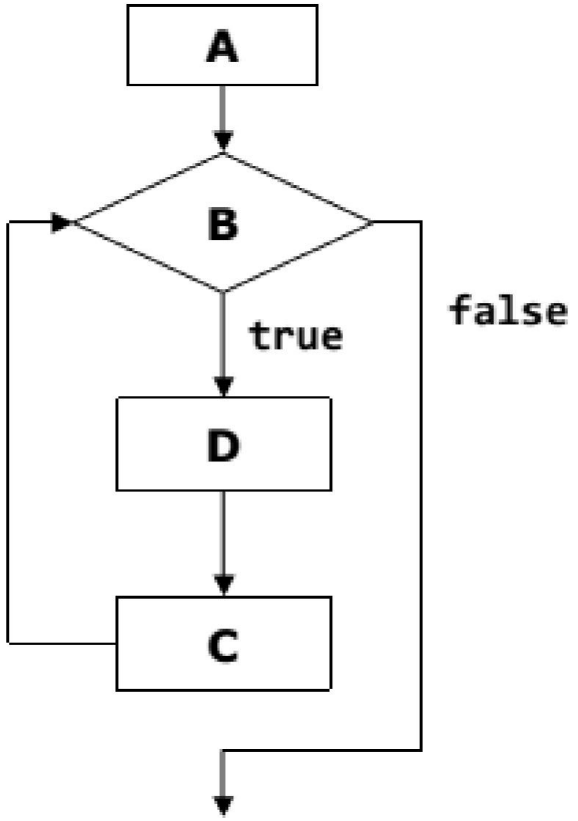
n = 2
m = 6
Product [n...m] = 720

For loop

```
for (initialization; condition; update)
{
   loop's body;
}
```

```
for (A; B; C)
{
   D;
}
```

```
for (int i=0; i<10; i++)
{
   /* loop body */
}
```



For loop

```
for (int i = 0; i <= 10; i++)
{
   Console.Write(i + " ");
}
```

```
for (int i = 1, sum = 1; i <= 128; i = i * 2, sum += i)
{
   Console.WriteLine("i={0}, sum={1}", i, sum);
}
```

For loop

```
for (int i = 0; i <= 10; i++)
{
    Console.Write(i + " ");
}
```

```
0 1 2 3 4 5 6 7 8 9 10
```

```
i=1, sum=1
i=2, sum=3
i=4, sum=7
i=8, sum=15
i=16, sum=31
i=32, sum=63
i=64, sum=127
i=128, sum=255
```

For loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());
decimal result = 1;
for (int i = 0; i < m; i++)
{
   result *= n;
}
Console.WriteLine("n^m = " + result);
```

For loop

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());
decimal result = 1;
for (int i = 0; i < m; i++)
{
    result *= n;
```

```
n = 2
m = 10
n^m = 1024
```

For loop

```
for (int small=1, large=10; small<large; small++, large--)
{
   Console.WriteLine(small + " " + large);
}
```

For loop

```
for (int small=1, large=10; small<large; small++, large--)
{
   Console.WriteLine(small + " " + large);
}
```

```
10 1
 9 2
8 3
7 4
6 5
```

For loop

```
int n = int.Parse(Console.ReadLine());
int sum = 0;
for (int i = 1; i <= n; i += 2)
{
   if (i % 7 == 0)
   {
      continue;
   }
   sum += i;
}
Console.WriteLine("sum = " + sum);
```

For loop

```
int n = int.Parse(Console.ReadLine());
int sum = 0;
for (int i = 1; i <= n; i += 2)
{
   if (i % 7 == 0)
   {
      continue;
   }
```

```
11
sum = 29
```

Foreach loop

```
foreach (type variable in collection)
{
   statements;
}
```

```
int[] numbers = { 2, 3, 5, 7, 11, 13, 17, 19 };
foreach (int i in numbers)
{
   Console.Write(" " + i);
}
Console.WriteLine();
string[] towns = { "London", "Paris", "Milan", "New York" };
foreach (string town in towns)
{
   Console.Write(" " + town);
}
```

Foreach loop

```
foreach (type variable in collection)
{
   statements;
}
```

```
int[] numbers = { 2, 3, 5, 7, 11, 13, 17, 19 };
foreach (int i in numbers)
```

2 3 5 7 11 13 17 19
London Paris Milan New York

Nested Loops

```
int n = int.Parse(Console.ReadLine());
for (int row = 1; row <= n; row++)
{
   for (int col = 1; col <= row; col++)
   {
      Console.Write(col + " ");
   }
   Console.WriteLine();
}
```

Nested Loops

```
int n = int.Parse(Console.ReadLine());
for (int row = 1; row <= n; row++)
{
   for (int col = 1; col <= row; col++)
   {
      Console.Write(col + " ");
   }
   Console.WriteLine();
}
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
```

Nested Loops

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());

for (int num = n; num <= m; num++)
{
   bool prime = true;
   int divider = 2;
   int maxDivider = (int)Math.Sqrt(num);
   while (divider <= maxDivider)
   {
      if (num % divider == 0)
      {
         prime = false;
         break;
      }
      divider++;
   }
   if (prime)
   {
      Console.Write(" " + num);
   }
}
```

Nested Loops

```
Console.Write("n = ");
int n = int.Parse(Console.ReadLine());
Console.Write("m = ");
int m = int.Parse(Console.ReadLine());

for (int num = n; num <= m; num++)
{
   bool prime = true;
   int divider = 2;
   int maxDivider = (int)Math.Sqrt(num);
   while (divider <= maxDivider)
   {
   }
}
```

```
n = 3
m = 75
3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
```