lesson 34 Bitwise operator AND

عندما نتعلم البرمجة فإننا نتعلم شيئين :

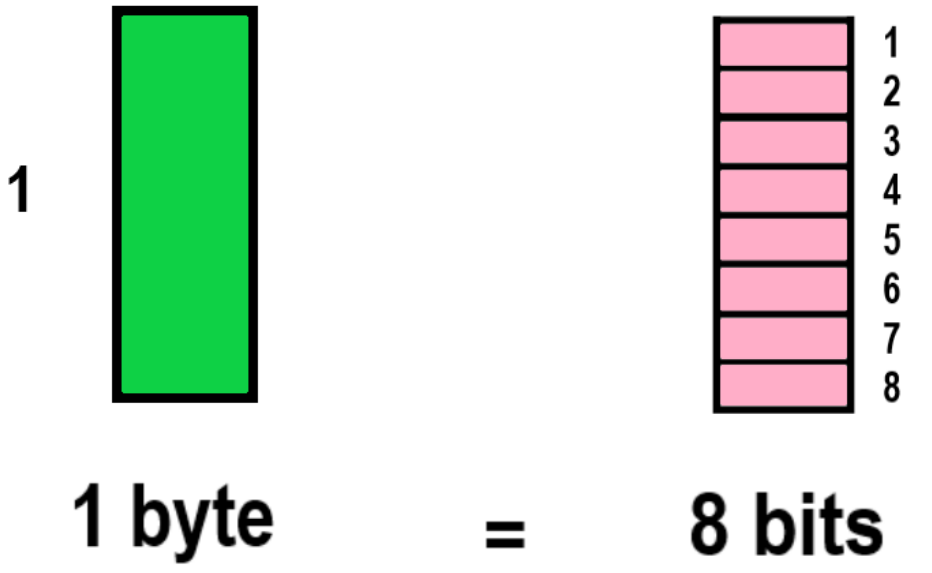
- الاول الأوامر البرمجية
- الثانى مهارة استخدام هذه الأوامر لعمل برنامج ما

فى هذا الكتاب ندرس أوامر لغة c، لكن لمعرفة استخدام تلك الأوامر بطريقة أمثل و نقوم بعمل برنامج سريع وقوى وليس به أخطاء لابد أن ندرس **Algorithms**
ندرس فى هذا الدرس شئ من الصعب أن نذكر عليه مثال فى استخدامه ، لكن استخدامه سيظهر عند دراسة ال **Algorithms**

أى رقم لدينا عبارة عن :

Bit – byte

ال byte يتكون من 8 bit



و عرفنا أيضا أن ال int يحجز مساحة 4 bytes في الذاكرة، و بم أن ال byte يحتوى على 8 bits إذا ال int يحتوى على 32 (4 × 8 bits) .

ال bit هي وحدة تخزين تستطيع أن تحمل 0 , 1 فقط، أي أن الأرقام و الحروف و الرموز و الصور و الألوان و مقاطع الفيديو و الموسيقى و الألعاب و كل ما نستطيع تخزينه في الكمبيوتر يتحول إلى 0 , 1 لكى يفهمه الكمبيوتر و يستطيع التعامل معه و تخزينه، و يتم ترجمة ال 0 , 1 إلى الأنواع المختلفة من البيانات عند استدعائها.

و بم أن الارقام بالنسبة للكمبيوتر شكلين الصفر والواحد عندما نقوم بحجز int :

```
int x;
```

يتم ترجمته إلى **32** رقم بجانب بعضهم متكونين من 0s , 1s
وعلى سبيل المثال رقم **5** بالنسبة للكمبيوتر يكون كالتالى :

1 0 1

لكن هو يتم تخزينه كالتالى فى الكمبيوتر:

00000000000000000000000000000101

لان **int** عبارة عن **32** رقم مقسمين الى **4** أجزاء كل جزء يحتوي على **8** أرقام يسمى **byte**

int =4 byte

كل جزء من 32 رقم يسمى **bit** كل digit يسمى **bit**

int = 32 bit

Byte = 8 bit

bitwise

من هذا الاسم يتضح لنا أنها تتعامل مع ال **bit**

إذا كان عندنا

```
int x = 5, y = 6;
int z = x & y;
```

ما معنى أن يكون z = x & y ؟

لإجراء عملية ال & بين رقمين، أولا نقوم بتحويل الرقمين إلى النظام ال binary
ثم إجراء عملية ال & بينهم .

طريقة عمل علامة **&** :

إذا كان هناك 1 و 1 متقابلان هذا معناه **true** إذا يكون الناتج (1)
إذا كان هناك 1 و 0 متقابلان هذا معناه **false** إذا يكون الناتج (0)
إذا كان هناك 0 و 0 متقابلان هذا معناه **false** إذا يكون الناتج (0)

	64	32	16	8	4	2	1
int x = 5	0	0	0	0	1	0	1
int y = 6	0	0	0	0	1	1	0
int z = x & y	0	0	0	0	1	0	0

الصف الثانى هو قيمة x ب **Binary**
الصف الثالث هو قيمة y ب **Binary**
الصف الرابع هو ناتج قيمة x&y
وبالتالى z = x & y تساوى 100 بنظام ال binary التي تساوى 4 بال decimal

كنا فى امر **if** نستخدم احياناً **&&** وهو معناه ان لابد ان يكون الشرطين **true** لتنفيذ امر **if**

```
if( true && true)
```

أما إذا كان احد الشرطين **false** يكون الناتج **false**

كذلك الأمر بالنسبة **&** في ال **bitwise** لكن نستخدم **and &** واحدة فقط وهنا تتعامل مع **bit** واحد فقط وليس قيمة x كلها

مثال أخر :

```
int x = 11, y = 3;
int z = x & y;
```

	64	32	16	8	4	2	1
int x = 11	0	0	0	1	0	1	1
int y = 3	0	0	0	0	0	1	1
int z = x & y	0	0	0	0	0	1	1

هنا الناتج سيكون 3 ، كما ذكرنا & تتعامل مع كل **bit**

True & True = True
True & false = False