lesson 38 CPP Tracing 1

عند العمل فى شركة، يوجد العديد من المبرمجين الذين يعملون على نفس البرنامج، وبالتالى كل مبرمج له جزئية خاص به، ولذلك لابد من تعلم **tracing** حتى تستطيع تحليل الكود ومعرفة ما الذي سيقوم بطباعته بدون الحاجة إلى عمل **run** للكود، لأنك لن تستطيع جمع كود البرنامج كله وعمل له **run** لاكتشاف الخطأ، ولذلك عليك تحليل الكود الخاص بك، ومعرفة الأخطاء إذا وجد

وهنا مثال :

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main() {
  int x = 5;
  // 5 هنا قمنا بتعريف متغير يساوى
  cout << x / 2 << endl;
  //  int/int سيكون الناتج 2 فقط لأنه
  int z = x * 100 / 2;
  // سيقوم بتنفيذ الضرب أولاً ثم القسمة
  //  5 * 100 / 2 = 250
  cout << z / 100 << "." << z % 100 << endl;
  // 2.50يكون ناتج الطباعة

  cout << (float)x / 2 << endl;
  //  casting سيكون الناتج 2.5  لأننا قمنا بعمل
  // وهو تحديد نوع المتغير الناتج
```

```
  int y = 2;
  float a = x / y;
  cout << a << endl;
  // int/int هنا سيكون الناتج 2 لأننا قمنا بقسمة

  a = 5.9;
  printf("%g\n", a);
  //%g  تعنى طباعة الرقم بدون اصفار بعد العلامة العشرية
  // 5.9 سيكون الناتج

  int b = 3111222333;
  printf("%d\n", b);
  // 2147483647 هو  int أكبر رقم موجب  يمكن تخزينه فى
  // -2147483648 هو  int أكبر رقم سالب  يمكن تخزينه فى
  // إذن هنا لا تستطيع طباعة الرقم موجب وسيتم طباعته رقم سالب
  //unsigned long long int ولحجز  هذا الرقم تقوم بحجزه من نوع
  // ليتم طباعته رقم موجب
  unsigned long long int c = 3111222333;
  printf("%llu \n", b);
  //%llu إثناء الطباعة نقوم بالاستخدام
}
```

output:

```
2
2.50
2.5
2
```

5.9
-1183744963
3111222333