الدرس 45 : (Lesson #45 - Stack vs Heap)

- الذاكرة **Memory** يقصد بها في البرمجة : **RAM**
- البرنامج يقسم في **RAM** الى **أربعة أقسام** هي

١. يخزن فيه كود الأوامر **Source Code / Instruction** (يأخذ مساحة قليلة)
٢. **Static / Global** هما **Variables** تكون دورة حياتهما طوال حياة كامل البرنامج (يأخذ مساحة قليلة)
٣. **Stack** هو: الذاكرة **Memory** التي يخصصها **Operating System** نظام الشغيل للبرنامج - بعد معرفة حجمه - (يأخذ مساحة على حسب البرنامج) يخزن فيه (Local : Variables / Function / **Pointers**)
٤. يستطيع Developer الوصول الى Heap بقية الذاكرة باستخدام **Pointers** (**Pointers** يخزن Address في Stack) أما الأشياء الديناميكية تخزن في **Heap** مثل

```
ptrX = new int;
```

Heap : **any dynamic** Variables / Objects / Arrays ...etc. يخزن فيه

الدرس 46 : Vectors (Lesson #46 - Access Elements)

- **Vector** يعامل معاملة **Array**

```
//  { 1, 2, 3, 4, 5 } Array تعامل معاملة
vector <int> num{ 1,2,3,4,5 };

// تستطيع الوصول الى أي عنصر بطريقتين

// NameVector.at(Index) الطريقة الأولى باستخدام
cout << "\n\n using .at(1) \n";
cout << "Element at Index 0 : " << num.at(0) << endl;
cout << "Element at Index 2 : " << num.at(2) << endl;
cout << "Element at Index 4 : " << num.at(4) << endl;
// Vector زيادة Index عن العناصر يعترض البرنامج على ذلك Exception ويذهب بك الى كود
cout << "Element at Index 5 : " << num.at(5) << endl;

// NameVector[Index] الطريقة الثانية باستخدام
cout << "\n\n using [1] \n";
cout << "Element at Index 0 : " << num[0] << endl;
cout << "Element at Index 2 : " << num[2] << endl;
cout << "Element at Index 4 : " << num[4] << endl;
// Garbage أو Wrong عند زيادة Index عن العناصر يحدث خطأ للبرنامج
cout << "Element at Index 5 : " << num[5] << endl;
```