

# HW4

**Write three functions in C or C++: one that declares a large array statically, one that declares the same large array on the stack, and one that creates the same large array from the heap. Call each of the subprograms a large number of times (at least 100,000) and output the time required by each. Explain the results.**

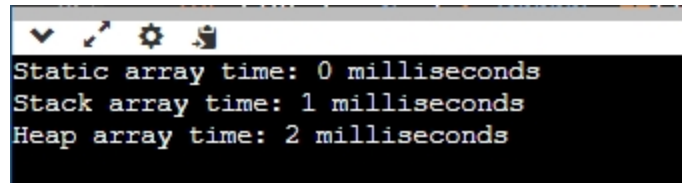
C++

```

1  #include <iostream>
2  #include <chrono>
3
4  using namespace std;
5  using namespace std::chrono;
6
7  void staticArray() {
8      static int arr[1024 * 10];
9  }
10
11 void stackArray() {
12     int arr[1024 * 10];
13 }
14
15 void heapArray() {
16     int* arr = new int[1024 * 10];
17     delete[] arr;
18 }
19
20 int main() {
21     auto start = high_resolution_clock::now();
22     for (int i = 0; i < 100000; ++i) {
23         staticArray();
24     }
25     auto stop = high_resolution_clock::now();
26     auto duration = duration_cast<milliseconds>(stop - start);
27     cout << "Static array time: " << duration.count() << " milliseconds" << endl;
28
29     start = high_resolution_clock::now();
30     for (int i = 0; i < 100000; ++i) {
31         stackArray();
32     }
33     stop = high_resolution_clock::now();
34     duration = duration_cast<milliseconds>(stop - start);
35     cout << "Stack array time: " << duration.count() << " milliseconds" << endl;
36
37     start = high_resolution_clock::now();
38     for (int i = 0; i < 100000; ++i) {
39         heapArray();
40     }
41     stop = high_resolution_clock::now();
42     duration = duration_cast<milliseconds>(stop - start);
43     cout << "Heap array time: " << duration.count() << " milliseconds" << endl;
44
45     return 0;
46 }
47

```

output

A screenshot of a terminal window with a dark background and light-colored text. The terminal shows three lines of output: 'Static array time: 0 milliseconds', 'Stack array time: 1 milliseconds', and 'Heap array time: 2 milliseconds'. Above the text, there are several small icons: a checkmark, a cursor, a gear, and a trash can.

```
Static array time: 0 milliseconds
Stack array time: 1 milliseconds
Heap array time: 2 milliseconds
```

Explain the results

**Static** : การประกาศอาร์เรย์แบบ static ในฟังก์ชัน staticArray() ที่บรรทัดที่ 7 นั้นมีความไวที่สุด เนื่องจาก การจัดสรรหน่วยความจำเกิดขึ้นในขั้นตอนของการโหลดโปรแกรม ซึ่งหมายความว่า อาร์เรย์จะถูกสร้างขึ้นและจัดเก็บในหน่วยความจำที่ถูกกำหนดไว้ตั้งแต่เริ่มต้นของโปรแกรม ซึ่งทำให้ ไม่มีความล่าช้าในการจัดสรรหน่วยความจำเมื่อฟังก์ชันถูกเรียกใช้ สิ่งนี้ช่วยลดการใช้งานทรัพยากร และเพิ่มประสิทธิภาพในระหว่างการทำงาน

**Stack** : การประกาศอาร์เรย์แบบ stack ในฟังก์ชัน stackArray() ที่บรรทัดที่ 11 นั้นมีความไวรองลงมาเนื่องจากการสร้าง array แบบ stack นั้นถูกสร้างขึ้นมาในระหว่าง run time และเป็นการ จัดการหน่วยความจำแบบ Dynamic

**Stack** : การประกาศอาร์เรย์แบบ heap ในฟังก์ชัน heapArray() ที่บรรทัดที่ 15 นั้นมีความไวต่ำ ที่สุดเนื่องจากการสร้าง array แบบ heap นั้นมีความซับซ้อนยิ่งกว่า โดยการจัดสรรของระบบในช่วง runtime