C++ Programming

Dynamic Memory Allocation

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

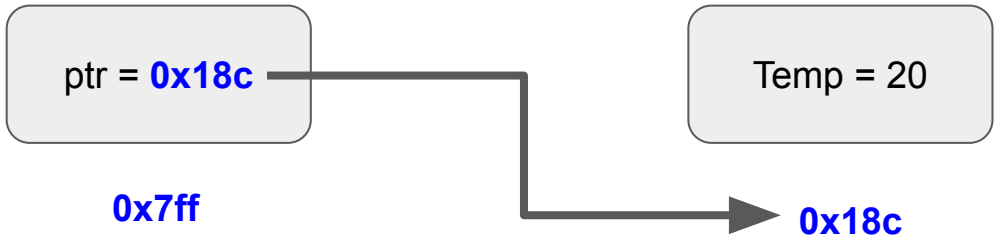
Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Arrays

- Recall we define array as following:
  - const int SIZE = 100;
  - int arr[SIZE];
- Once declared, we can’t change its size
  - In other words: Fixed allocated memory!
- In practice: we don’t know the needed size?
  - Workaround: Define maximum possible size?
  - Not practical!
- Pointers is our way to allocate **dynamic** memory

Creating a single element

```
int x = 10;
int *p0 = &x;
int *p1 {nullptr};

int *p2 = new int;  // Dynamic Allocation
*p2 = 20;           // set value

// 20 0x18c 0x7ff
cout<<*p2<<" "<<p2<<" "<<&p2<<"\n";

int *p3 = new int {30}; // C++11 brace initialization syntax
//int *p3 = new int (30);   // initializer

// p0 points to X. never delete
// p1 is nullptr. never delete

cout<<*p2<<" "<<*p3<<"\n";

// We created these 2 dynamically. Delete them
delete p2;
delete p3;
p2 = p3 = nullptr;
```



Creating more than element

```
4  int main() {
5
6      const int SIZE = 10;
7      int arr1[SIZE] {0};
8
9      int n = 20; // NOT const
10     int *pArr = new int [n];
11
12     for (int i = 0; i < n; ++i)
13         pArr[i] = i;
14
15     delete[] pArr;  // free the array
16
17     // WRONG and will compile
18     //delete pArr;
19
20     return 0;
21 }
22
```

Why delete?

- new & delete are operators to create/release memory
- When we use new: some part of RAM is reserved for you
- If you did not delete?
  - The computer never gets a request to release this reserved RAM
- What if you forgot to delete?
  - **Memory leak**: A part of memory that is never released (till a machine restart)
  - If there is a function that has a big memory leak and is called a lot?
    - Whole computer memory RAM is reserved
    - The machine **hangs**
  - Side tips:
    - There are tools to discover memory leaks
    - Checking your task manager might reveal some cases

Common Mistakes

- Use delete instead of delete []
- Use delete[] instead of delete
- Delete some pointer **twice** = **Dangling** pointer
- Accessing a deleted memory
- Accessing uninitialized variable
- Tip:
  - Never leave pointer uninitialized. At least to nullptr
  - Whenever delete a pointer: assign to nullptr
  - Before deleting a pointer: Think in the above mistakes

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."