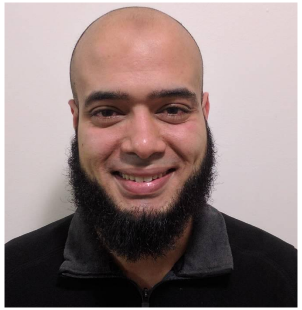
C++ Programming Reference

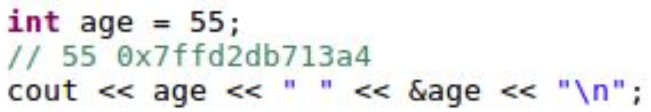
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

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



Recall the memory

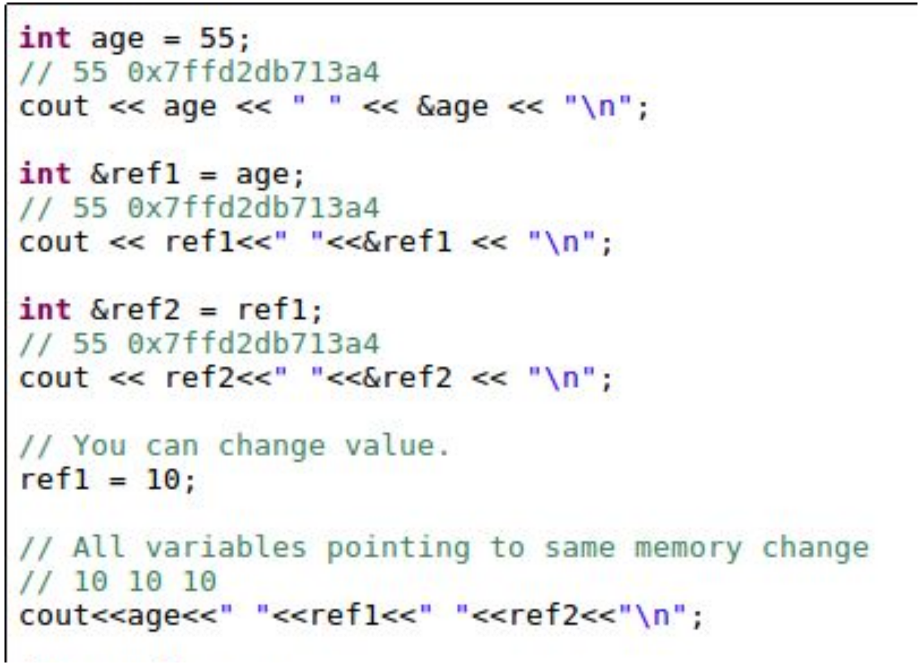
- When you declare a variable int age {55};
  - A memory location is reserved for it
  - E.g. Location 32714
    - We can represent in hexadecimal representation 0x7fca (online calculator)
- We can use **&** to get the address



```
int age = 55;
// 55 0x7ffd2db713a4
cout << age << " " << &age << "\n";
```

Location	Name/Value	Data Type
12714	**Weight = 92.5**	Double
32714 (**0x7ffcca**)	**Age = 55**	int
34014	**Gender = Male**	Boolean
35714	**Name = “Mostafa”**	String

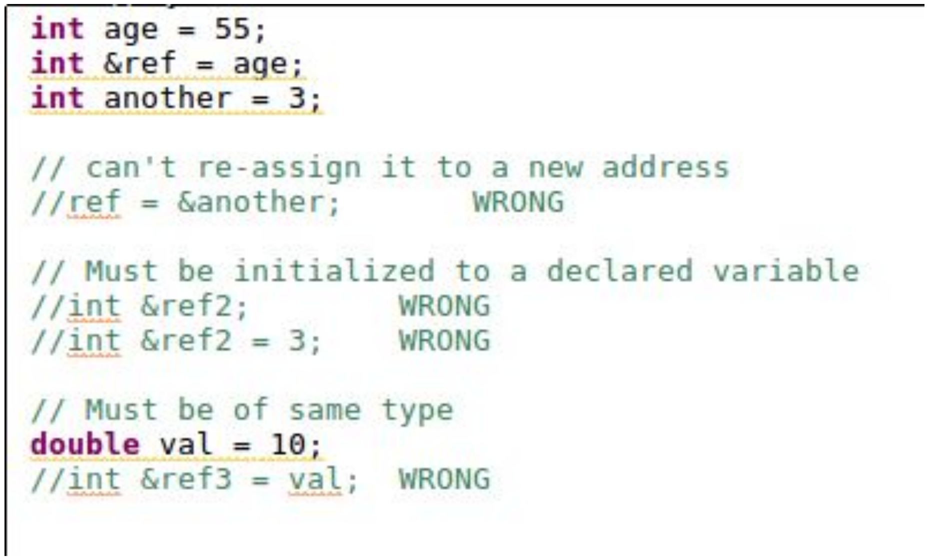
Reference



```
int age = 55;
// 55 0x7ffd2db713a4
cout << age << " " << &age << "\n";

int &ref1 = age;
// 55 0x7ffd2db713a4
cout << ref1<<" "<<&ref1 << "\n";

int &ref2 = ref1;
// 55 0x7ffd2db713a4
cout << ref2<<" "<<&ref2 << "\n";

// You can change value.
ref1 = 10;

// All variables pointing to same memory change
// 10 10 10
cout<<age<<" "<<ref1<<" "<<ref2<<"\n";
```

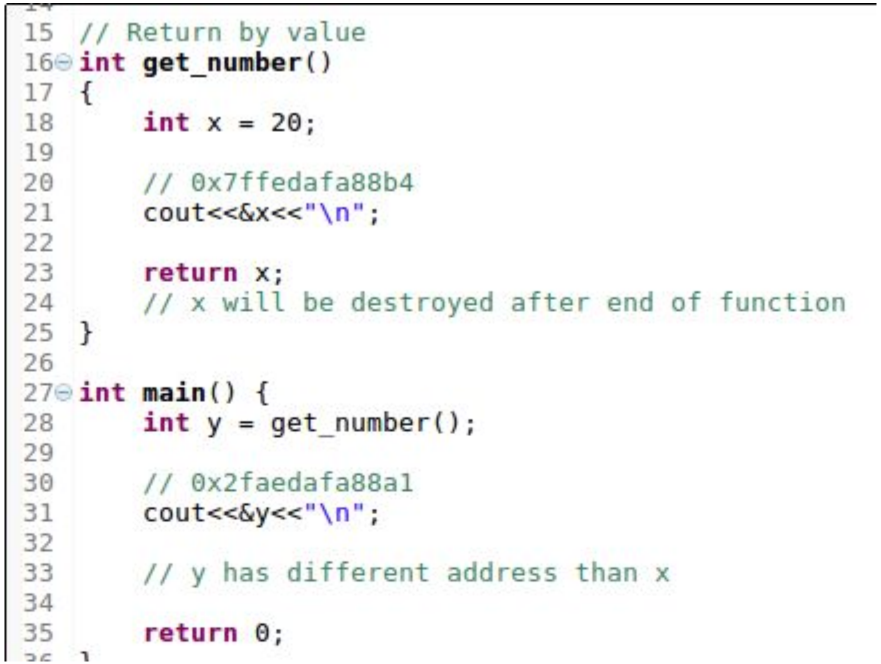
- & ⇒ address-of operator
- A reference is declared as an **alias** of a variable.
- It **stores** the **address** of the variable

Reference: Constraints



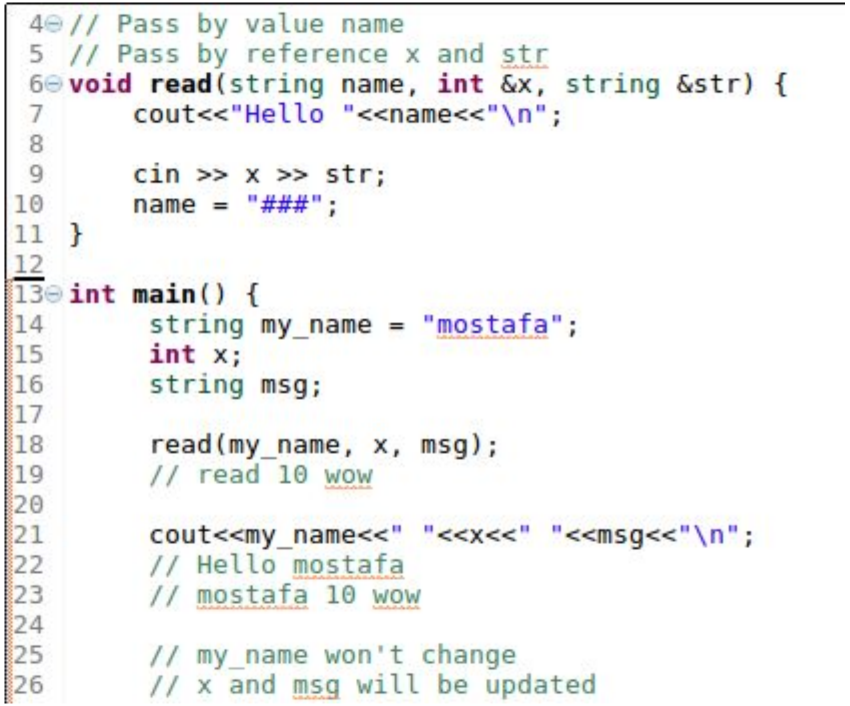
```
int age = 55;
int &ref = age;
int another = 3;

// can't re-assign it to a new address
//ref = &another;        WRONG

// Must be initialized to a declared variable
//int &ref2;         WRONG
//int &ref2 = 3;     WRONG

// Must be of same type
double val = 10;
//int &ref3 = val;  WRONG
```

Return by value



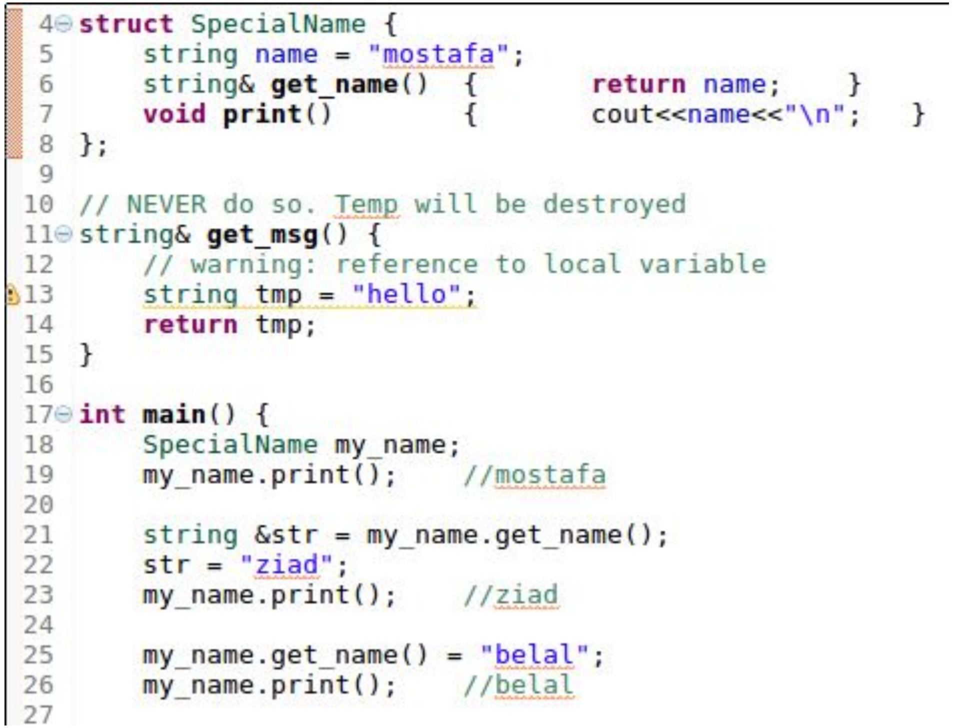
```
// Return by value
int get_number()
{
    int x = 20;

    // 0x7ffedafa88b4
    cout<<&x<<"\n";

    return x;
    // x will be destroyed after end of function
}

int main() {
    int y = get_number();

    // 0x2faedafa88a1
    cout<<&y<<"\n";

    // y has different address than x

    return 0;
```

Pass by reference



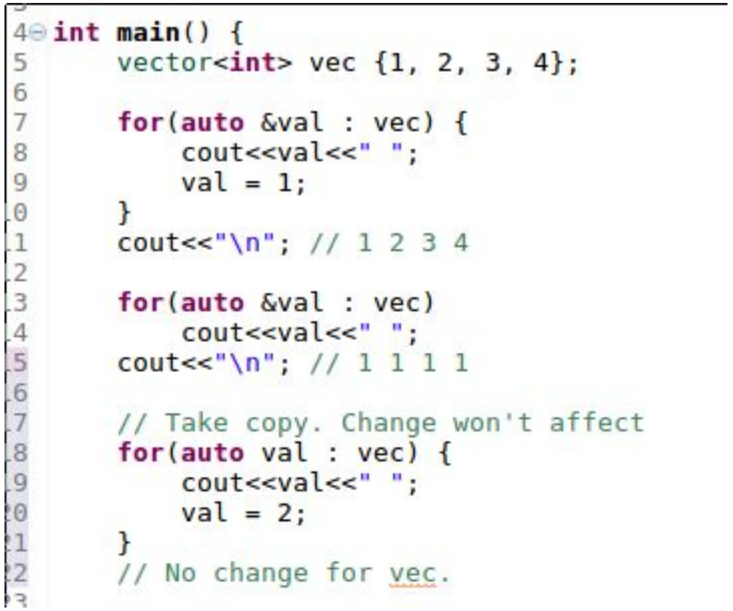
```
// Pass by value name
// Pass by reference x and str
void read(string name, int &x, string &str) {
    cout<<"Hello "<<name<<"\n";

    cin >> x >> str;
    name = "###";
}

int main() {
     string my_name = "mostafa";
     int x;
     string msg;

     read(my_name, x, msg);
     // read 10 wow

     cout<<my_name<<" "<<x<<" "<<msg<<"\n";
     // Hello mostafa
     // mostafa 10 wow

     // my_name won't change
     // x and msg will be updated
```

Return by reference



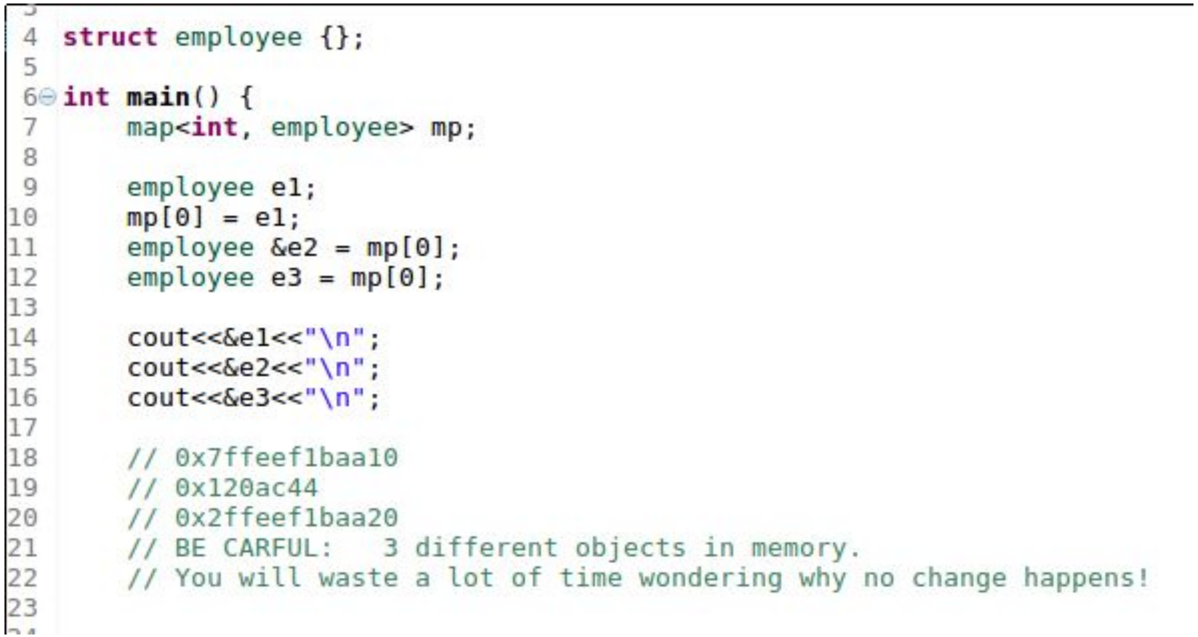
```
struct SpecialName {
    string name = "mostafa";
    string& get_name()  {       return name;     }
    void print()        {       cout<<name<<"\n";   }
};

// NEVER do so. Temp will be destroyed
string& get_msg() {
    // warning: reference to local variable
    string tmp = "hello";
    return tmp;
}

int main() {
    SpecialName my_name;
    my_name.print();    //mostafa

    string &str = my_name.get_name();
    str = "ziad";
    my_name.print();    //ziad

    my_name.get_name() = "belal";
    my_name.print();    //belal

```

Iterate using the reference



```
int main() {
    vector<int> vec {1, 2, 3, 4};

    for(auto &val : vec) {
        cout<<val<<" ";
        val = 1;
    }
    cout<<"\n"; // 1 2 3 4

    for(auto &val : vec)
        cout<<val<<" ";
    cout<<"\n"; // 1 1 1 1

    // Take copy. Change won't affect
    for(auto val : vec) {
        cout<<val<<" ";
        val = 2;
    }
    // No change for vec.
```

Common Mistake



```
struct employee {};

int main() {
    map<int, employee> mp;

    employee e1;
    mp[0] = e1;
    employee &e2 = mp[0];
    employee e3 = mp[0];

    cout<<&e1<<"\n";
    cout<<&e2<<"\n";
    cout<<&e3<<"\n";

    // 0x7ffeef1baa10
    // 0x120ac44
    // 0x2ffeef1baa20
    // BE CARFUL:   3 different objects in memory.
    // You will waste a lot of time wondering why no change happens!

```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."