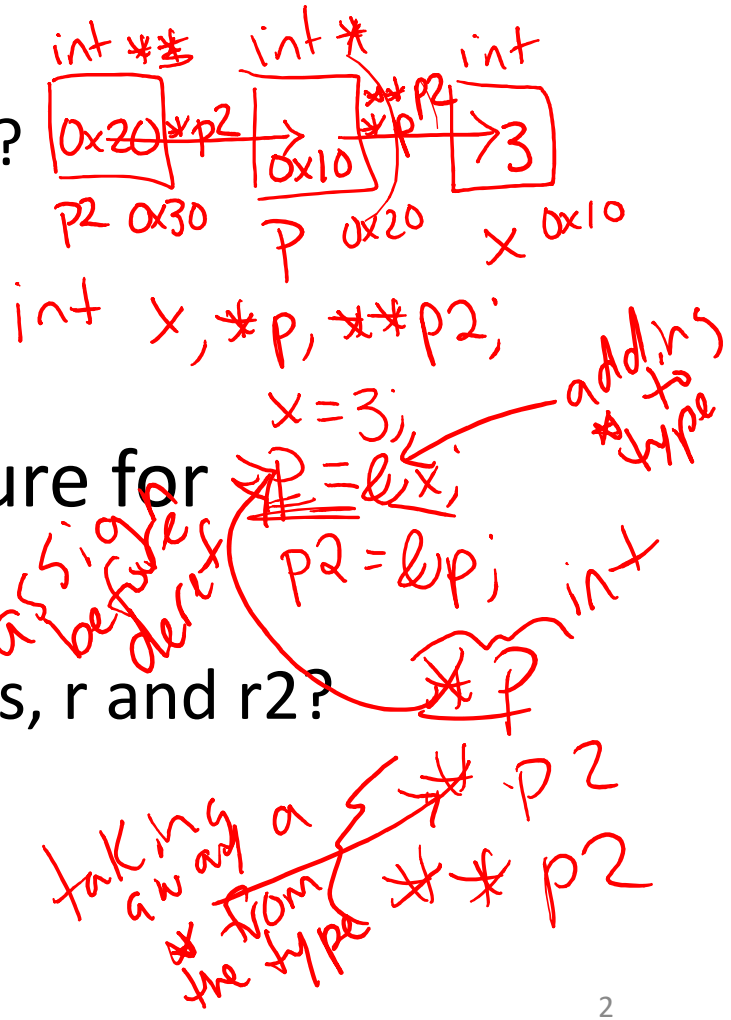
CS 161
Intro to CS I

Stack vs. Heap and 1-d Arrays

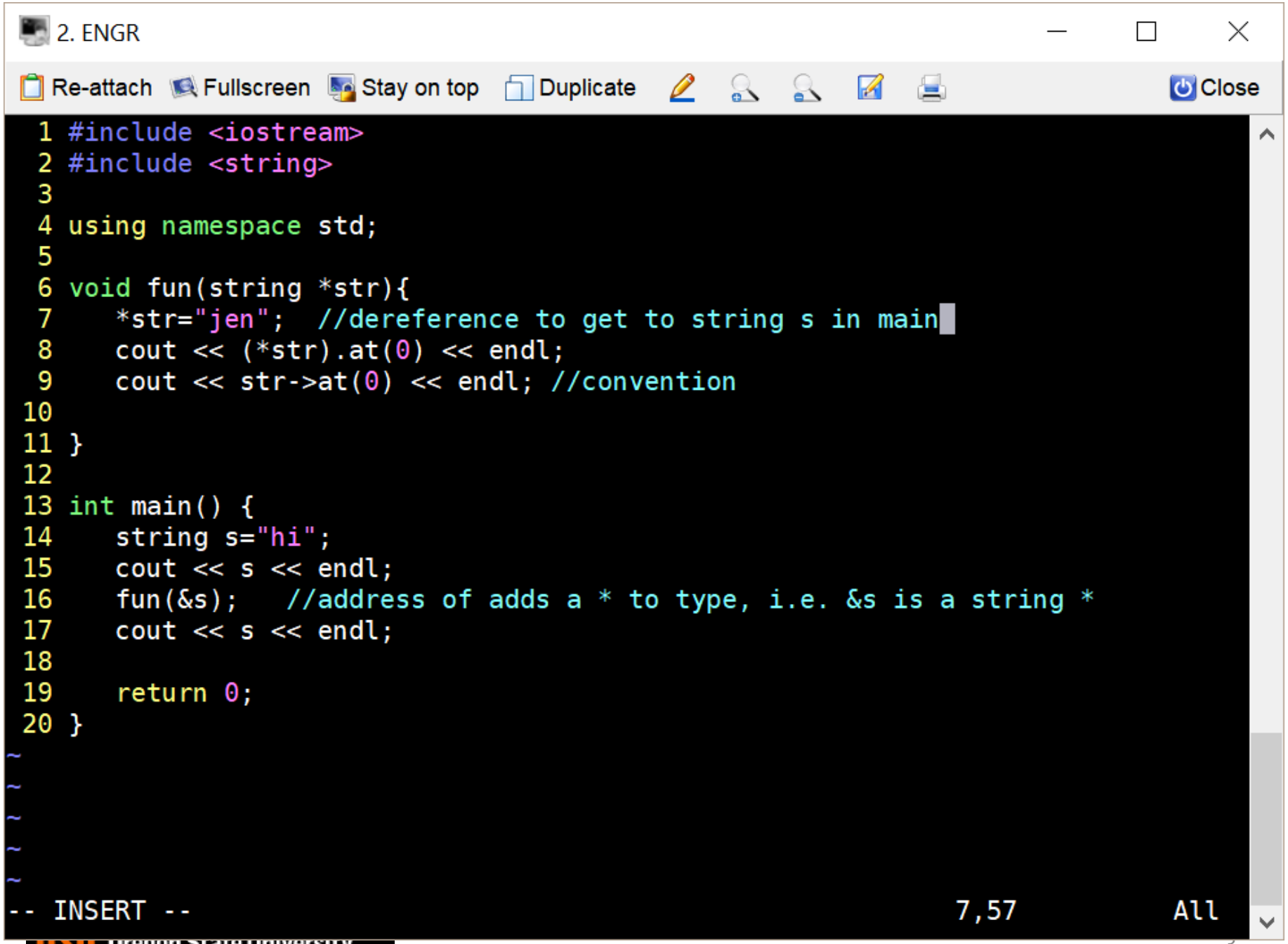
In-class Exercise
Pointers vs. References

- What if you made a pointer (p2) that points to a pointer (p) to an int (x)?
  - What would the picture look like?
  - Write the code for this picture.



- Can you make this same picture for references?
  - What if you had two references, r and r2?

NO


2. ENGR
Re-attach
Fullscreen
Stay on top
Duplicate
Close
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 void fun(string *str){
7 *str="jen"; //dereference to get to string s in main
8 cout << (*str).at(0) << endl;
9 cout << str->at(0) << endl; //convention
10
11 }
12
13 int main() {
14 string s="hi";
15 cout << s << endl;
16 fun(&s); //address of adds a * to type, i.e. &s is a string *
17 cout << s << endl;
18
19 return 0;
20 }
-- INSERT -- 7,57 All

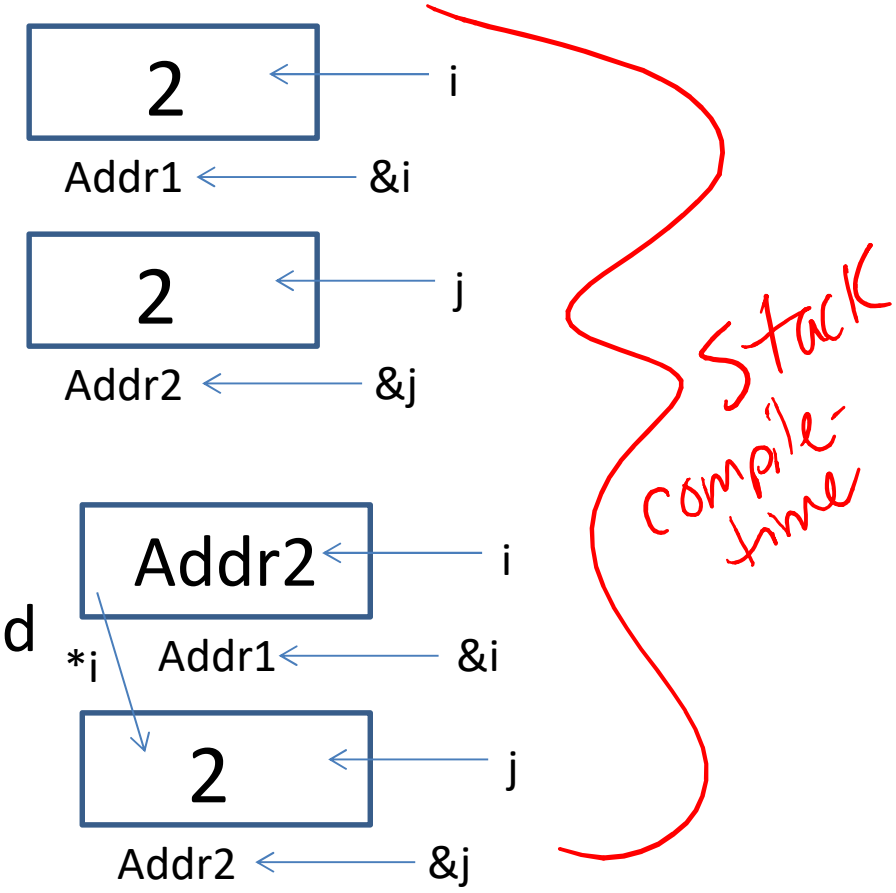
Revisit Variables vs. Pointers

- Value Semantics
  - Values stored directly
  - Copy of value is passed

```
int i, j=2;
i=j;
```

- Pointer Semantics
  - Address to variable is stored
  - Copy of address is passed

```
int *i, j=2;
i=&j;
```



What if we don't have the j?

- We need to create the address space.
- How do we do this?
  - **new** type;
- For example:

```
int *i;
i = new int;   //new returns an address
*i = 10;
```

heap
runtime

Binky Pointer Video

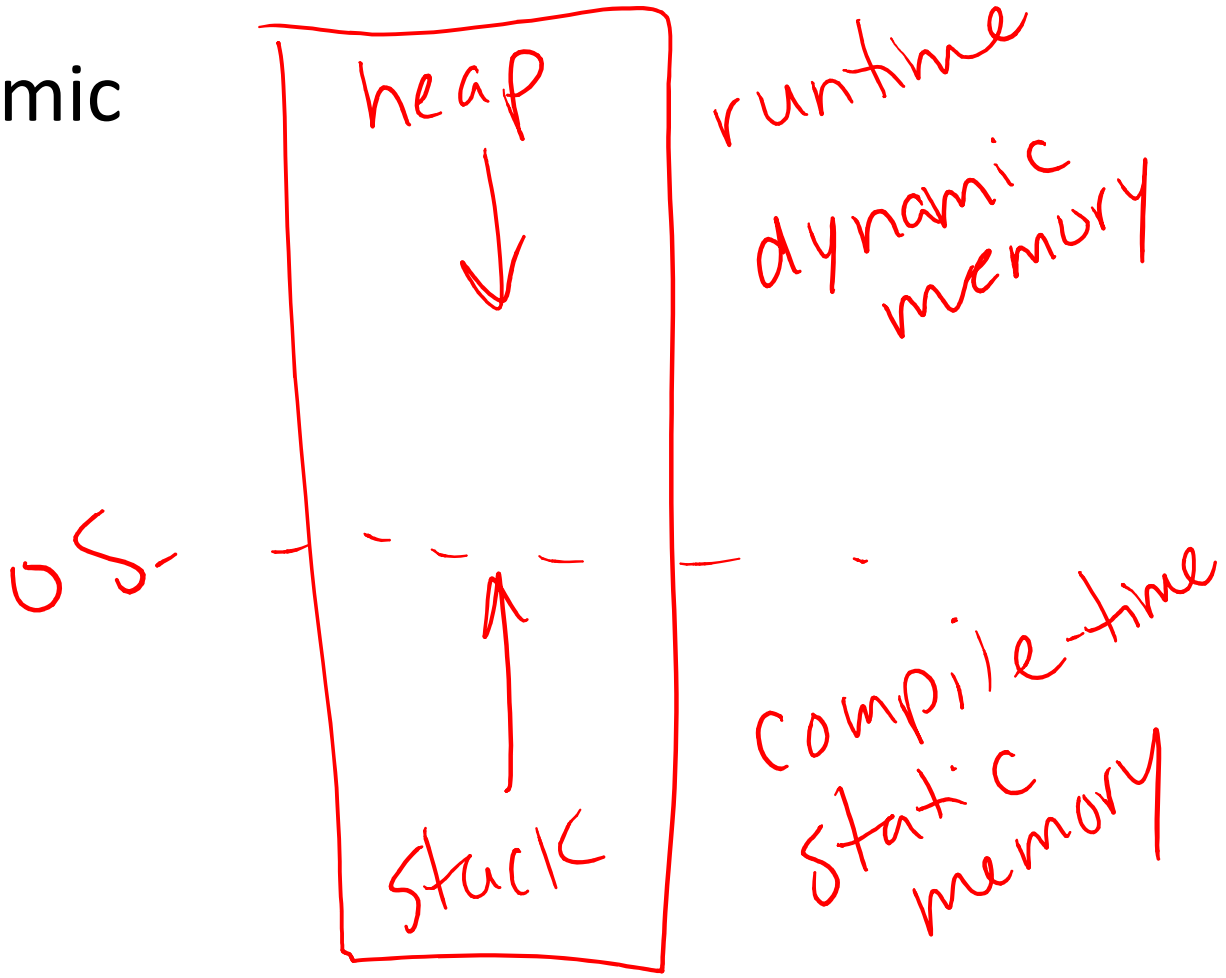
- Watch the C++ Stanford Binky video:

http://cslibrary.stanford.edu/104/

… and make sure you don't blow binky's head off in the future☺

Stack vs. Heap

- Static vs. Dynamic



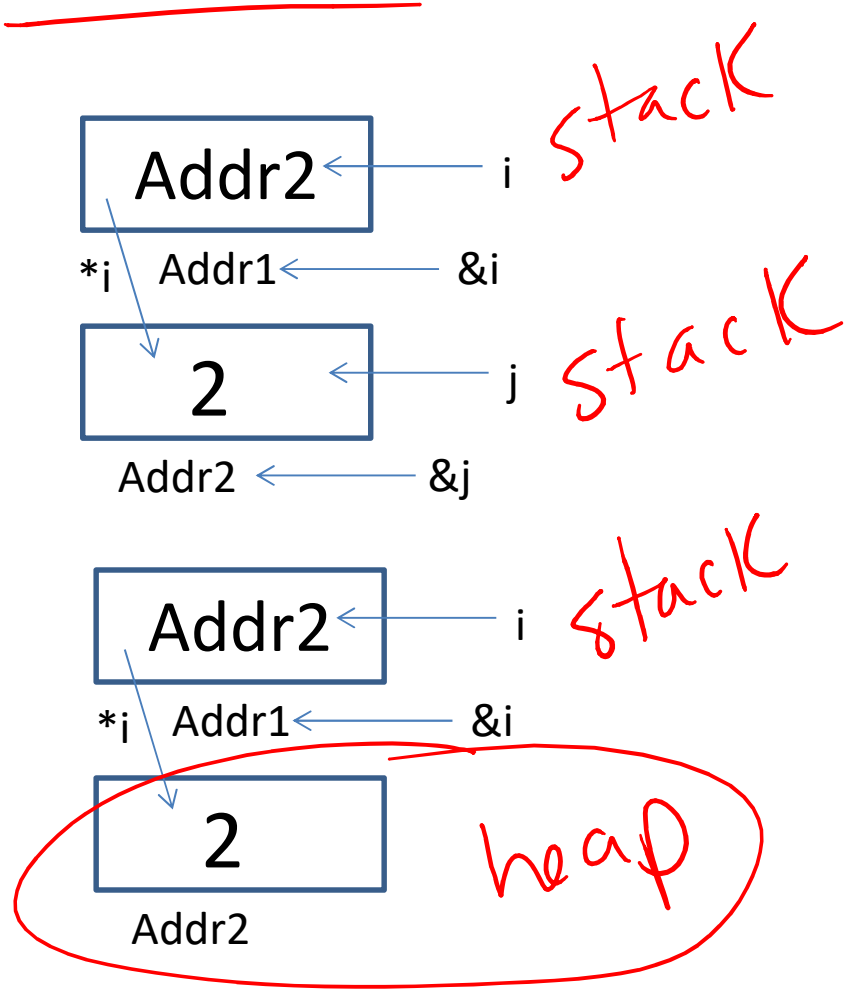
Static vs. Dynamic

- Static Semantics
  - Assign address of variable

```
int *i, j=2;
i=&j;
```

- Dynamic Semantics
  - Create memory
  - Assign memory to pointer

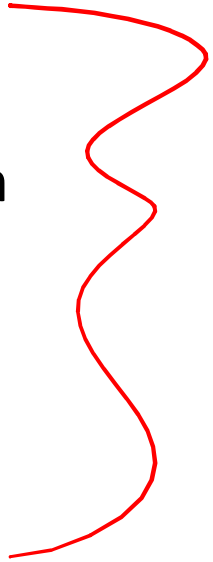
```
int *i=NULL;
i=new int;
*i=2;
```



What About Memory Leaks?

- What happens here…

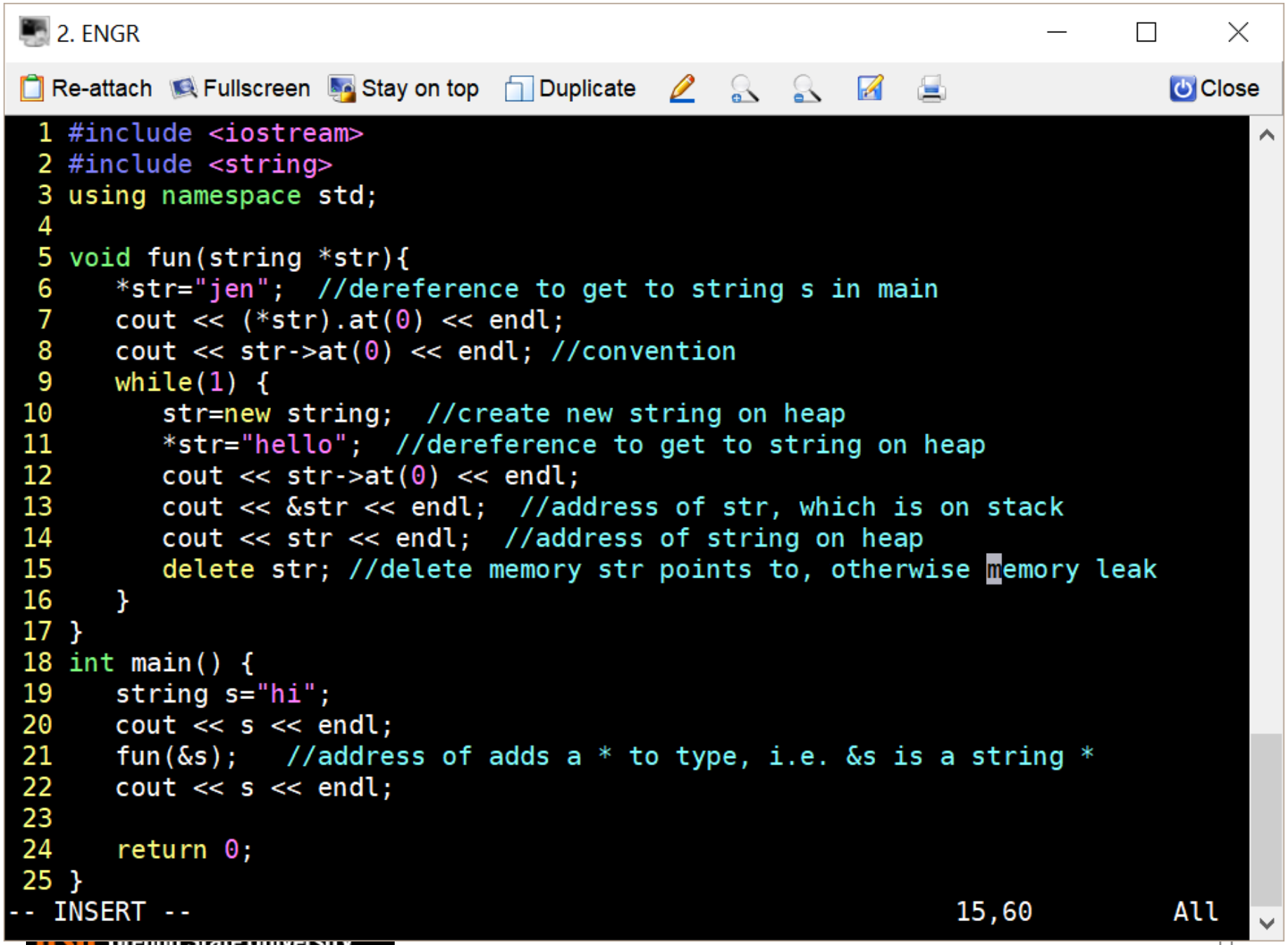
```
…
int main () {
  int *i=NULL;  //created in main function
  while(1) {
       i = new int;
    }
}
```



Fixing Memory Leaks…

- What happens here…

```
…
int main () {
  int *i=NULL;//created in main function
  while(1) {
    i = new int;
    delete i; //free memory that i points to, preventing mem leaks
  }
}
```

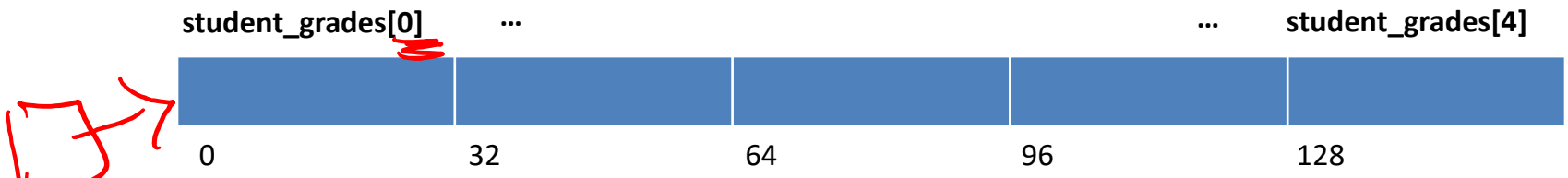

2. ENGR
Re-attach
Fullscreen
Stay on top
Duplicate
Close
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 void fun(string *str){
6 *str="jen"; //dereference to get to string s in main
7 cout << (*str).at(0) << endl;
8 cout << str->at(0) << endl; //convention
9 while(1) {
10 str=new string; //create new string on heap
11 *str="hello"; //dereference to get to string on heap
12 cout << str->at(0) << endl;
13 cout << &str << endl; //address of str, which is on stack
14 cout << str << endl; //address of string on heap
15 delete str; //delete memory str points to, otherwise memory leak
16 }
17 }
18 int main() {
19 string s="hi";
20 cout << s << endl;
21 fun(&s); //address of adds a * to type, i.e. &s is a string *
22 cout << s << endl;
23
24 return 0;
25 }
-- INSERT -- 15,60 All

What is an Array?

- **Array (ar·ray) *n*.** An ordered arrangement of related items.
  - Example: Array of colors in a rainbow.
    - Related items?
    - Ordered arrangement?
  - Class examples?
  - Computer Science
    - Same data type/data structure
    - Contiguous memory locations

Create 1-D Array

```
int student_grades[5];
```



- How do you access each item?
- What does the array name represent?
- Why is the array name the address of 1st element?
- What are the initial values?