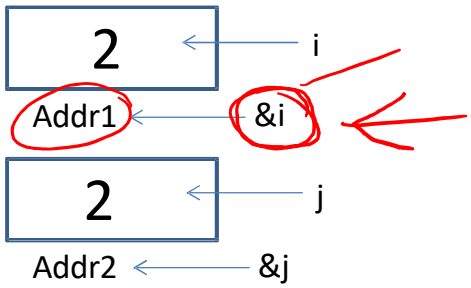
Variables vs. Pointers

Oregon State University
College of Engineering

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

  int i, j=2;

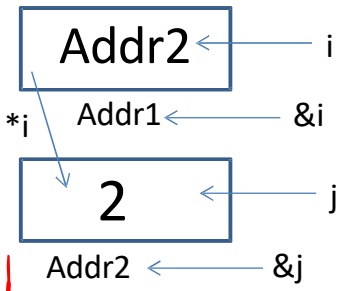
  i=j;



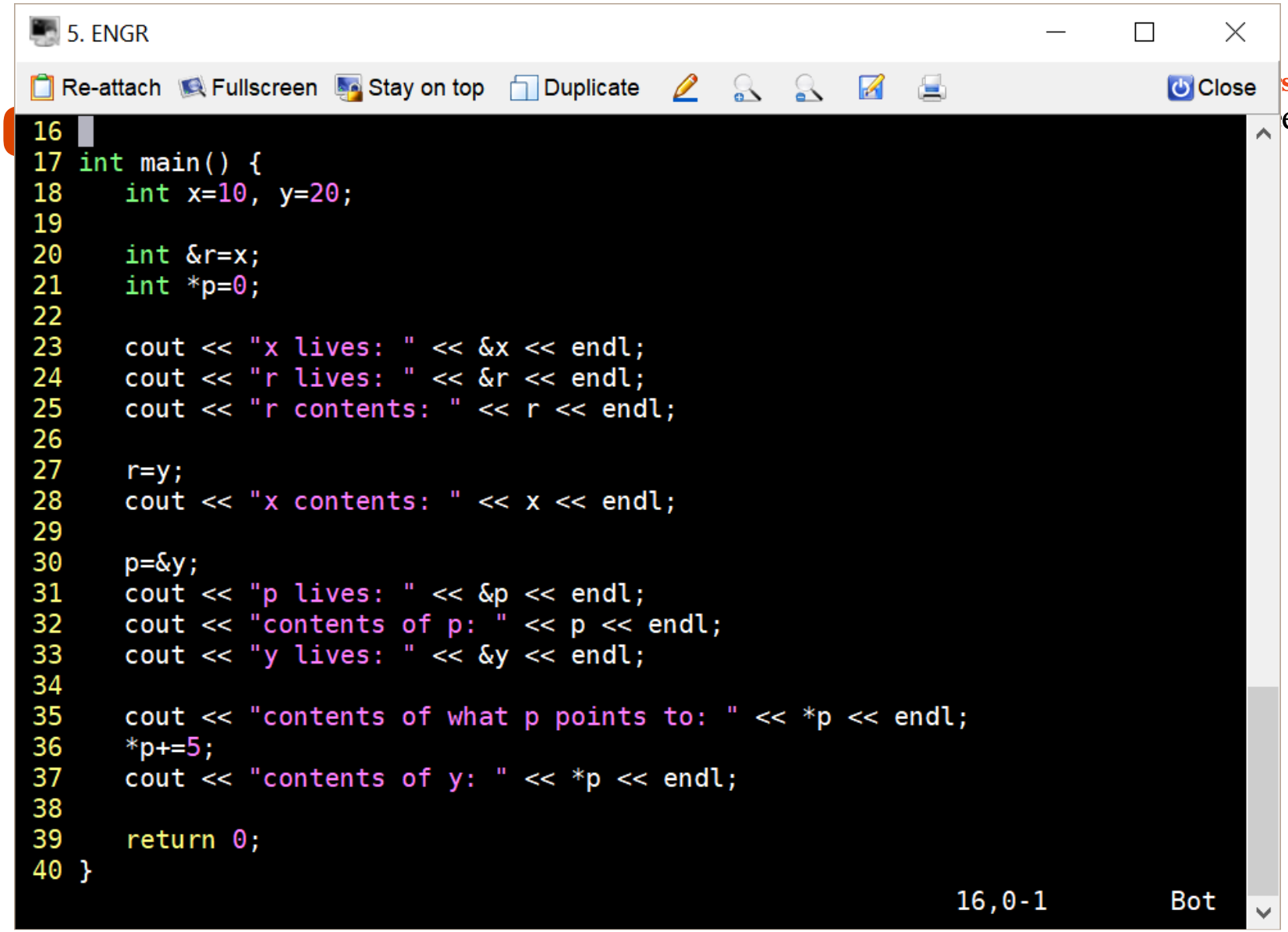
- Pointer Semantics
  - – Address to variable is stored
  - – Copy of address is passed

  int *i, j=2;

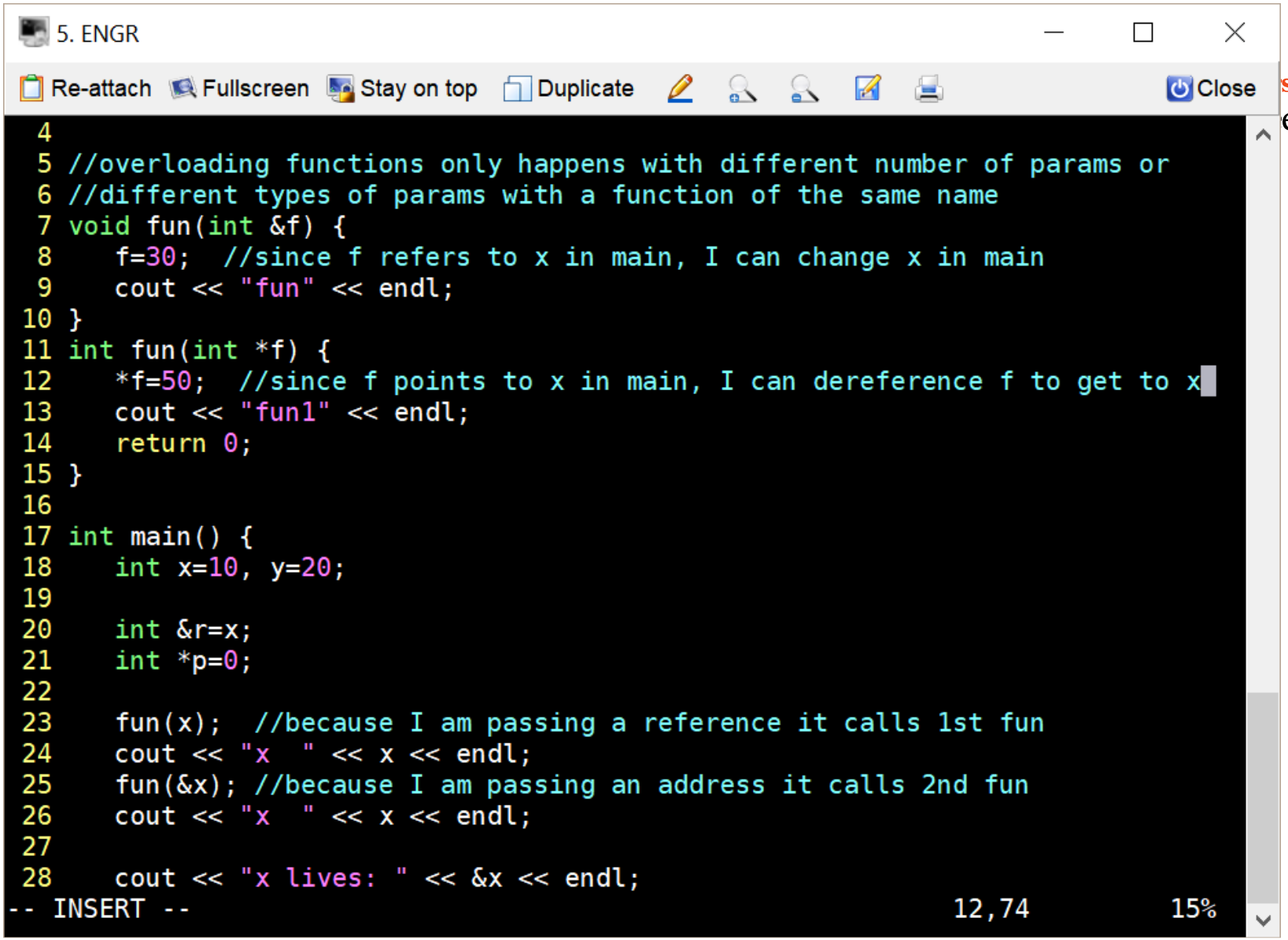
  i=&j;



int &x=j;


5. ENGR
Re-attach
Fullscreen
Stay on top
Duplicate
Close
16
17 int main() {
18 int x=10, y=20;
19
20 int &r=x;
21 int *p=0;
22
23 cout << "x lives: " << &x << endl;
24 cout << "r lives: " << &r << endl;
25 cout << "r contents: " << r << endl;
26
27 r=y;
28 cout << "x contents: " << x << endl;
29
30 p=&y;
31 cout << "p lives: " << &p << endl;
32 cout << "contents of p: " << p << endl;
33 cout << "y lives: " << &y << endl;
34
35 cout << "contents of what p points to: " << *p << endl;
36 *p+=5;
37 cout << "contents of y: " << *p << endl;
38
39 return 0;
40 }
16,0-1
Bot

sity
ering


5. ENGR
Re-attach
Fullscreen
Stay on top
Duplicate
Close
4
5 //overloading functions only happens with different number of params or
6 //different types of params with a function of the same name
7 void fun(int &f) {
8 f=30; //since f refers to x in main, I can change x in main
9 cout << "fun" << endl;
10 }
11 int fun(int *f) {
12 *f=50; //since f points to x in main, I can dereference f to get to x
13 cout << "fun1" << endl;
14 return 0;
15 }
16
17 int main() {
18 int x=10, y=20;
19
20 int &r=x;
21 int *p=0;
22
23 fun(x); //because I am passing a reference it calls 1st fun
24 cout << "x " << x << endl;
25 fun(&x); //because I am passing an address it calls 2nd fun
26 cout << "x " << x << endl;
27
28 cout << "x lives: " << &x << endl;
-- INSERT --
12,74
15%
sity
ering

Pointer and References Cheat Sheet

Oregon State University
College of Engineering

- *
  - If used **in a declaration** (which includes function parameters), it **creates** the pointer.
    - Ex. int *p; //p will hold an address to where an int is stored
  - If used **outside a declaration**, it **dereferences** the pointer
    - Ex. *p = 3; //**goes to the address** stored in p and stores a value
    - Ex. cout << *p; //**goes to the address** stored in p and fetches the value
- &
  - If used **in a declaration** (which includes function parameters), it **creates and initializes** the reference.
    - Ex. void fun(int &p); //p will refer to an argument that is an int by implicitly using *p (dereference) for p
    - Ex. int &p=a; //p will refer to an int, a, by implicitly using *p for p
  - If used **outside a declaration**, it means **"address of"**
    - Ex. p=&a; //**fetches the address of** a (only used as rvalue!!!) and store the address in p.