Ways C differs from Java

1/22/25

Which of the following lines will print the value of an int x?

A. printf(x);
B. printf(“%c\n”, x);
C. printf(“%d\n”, x);
D. printf(“%f\n”, x);
E. What’s an int?

Which of the following lines will print the value of an int x?

A. printf(x);
B. printf("%c\n", x);
C. printf("%d\n", x);
D. printf("%f\n", x);
E. What's an int?

Similarities between C and Java

Difference: 1-pass compilation

- Must declare functions before they are used (or compiler will guess at the type...)

```
void myFun();      //declaration = signature w/o body

//calls to the function work properly here

void myFun() {     //actual definition of the function
      ...
}
```

Difference: No boolean type

- Uses integers instead

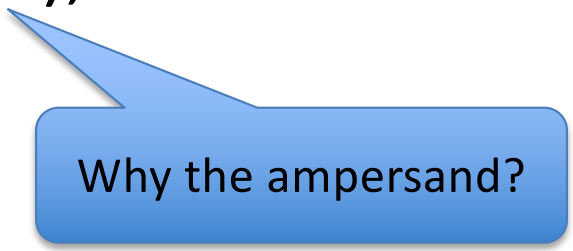
  0 = false

  Anything else = true

Difference: Pointers

- For input, use scanf, an analog of printf

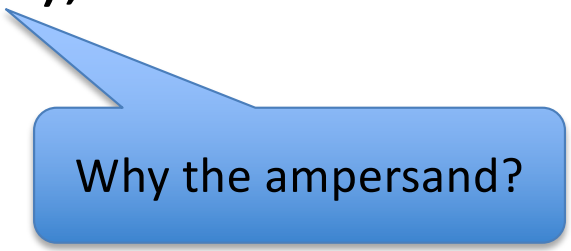
```
scanf("%d", &x);
```



Difference: Pointers

- For input, use scanf, an analog of printf

```
scanf("%d", &x);
```



- New operator in C
- Gets the memory address of a variable
  - scanf needs to put its value somewhere

Pointers

- Special kind of variable that stores a memory address

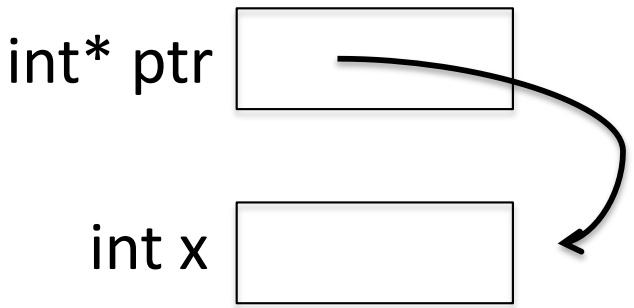
```
int x;      //x stores an int
int* p;     //p is a pointer to an integer
            //type is pronounced “int star”

p = &x;     //set using the ampersand

*p = 5;     //follow the pointer using * to access or
            //change the value to which it points
```

Pointers in pictures

- Variables that store an address in memory
- Get an address with &, use it with *



More about pointers

- Special value NULL (generally 0)
- Allowed to have pointers to pointers:

```
int x;
int* p = &x;
int** p2 = &p;
```

- Can assign pointers (copy addresses):

```
int* ptr1 = ...;
int* ptr2 = ptr1;
```

```
int x = 214;
int y = 305;
int* p = &x;
int* q = &y;
p = q;
```

Which of the following is NOT true after the above?

A. x has value 305
B. y has value 305
C. *p has value 305
D. *q has value 305
E. All of the above are true

```
int x = 214;
int y = 305;
int* p = &x;
int* q = &y;
p = q;
```

Which of the following is NOT true after the above?

A. x has value 305
B. y has value 305
C. *p has value 305
D. *q has value 305
E. All of the above are true

Multiple pointers and types

- Using & increases the level of indirection (another * and “pointer to”)
- Using * decreases it
- Example: If p is an int*:
  - *p is an int
  - &p is an int**

Which of the following is a problem with the code below?

```
int x = 214;            //line 1
int* p = &x;            //line 2
*p = 5;                 //line 3
int** q = &p;           //line 4
int* r = *q;            //line 5
```

A. Type error on line 2
B. Type error on line 3
C. Type error on line 4
D. Type error on line 5
E. The code is fine (albeit not useful)

Which of the following is a problem with the code below?

```
int x = 214;          //line 1
int* p = &x;          //line 2
*p = 5;               //line 3
int** q = &p;         //line 4
int* r = *q;          //line 5
```

A. Type error on line 2
B. Type error on line 3
C. Type error on line 4
D. Type error on line 5
E. The code is fine (albeit not useful)

Mystery functions!