

Output Parameters

*** Misuse of output parameters is the ***
 *** largest cause of errors in the course! ***

- ❖ Output parameter
 - a parameter that we use to store an output of a function.
 - can modify something in the caller's scope by dereferencing the pointer parameter.
 - Useful if you want to have multiple returns
- ❖ Caller passes in a pointer to the type they want.
 - If caller wants an `int`,
 - Declare an `int variable`
 - Pass in `&variable`
- ❖ Function dereferences the pointer parameter, and sets output.

```

void get333(int* output) {
    *output = 333;
}

int main(int argc, char** argv) {
    int num;
    get333(&num);
}
  
```

24

24

Poll Everywhere pollev.com/cse33320su

What would be the best way to invoke `genstrc.generateString()`?

```

void generateString(char** output);

int main(int argc, char** argv) {
    /* ??? */
    return EXIT_SUCCESS;
}
// ...
  
```

A. `char** result;`
`generateString(result);`
`printf(*result);`

B. `char* result[1] = {NULL};`
`generateString(result);`
`printf(result[0]);`

C. `char* str;`
`char** result = &str;`
`generateString(result);`
`printf(str);`

D. `char* result;`
`generateString(&result);`
`printf(result);`

E. We're lost...

25

25

Pointer Arithmetic

- ❖ Pointers are *typed*
 - Tells the compiler the size of the data you are pointing to
 - Exception: `void*` is a generic pointer (*i.e.* a placeholder)
- ❖ Pointer arithmetic is scaled by `sizeof(*p)`
 - Works nicely for arrays
 - Does not work on `void*`, since `void` doesn't have a size!
 - Not allowed, though confusingly GCC allows it as an extension 😊
- ❖ Valid pointer arithmetic:
 - Add/subtract an integer to/from a pointer
 - Subtract two pointers (within stack frame or malloc block)
 - Compare pointers (<, <=, ==, !=, >, >=), including NULL
 - ... but plenty of valid-but-inadvisable operations, too

31

31

Polling Question boxarrow2.c

```

int main(int argc, char** argv) {
    int arr[3] = {2, 3, 4};
    int* p = &arr[1];
    int** dp = &p; // pointer to a pointer

    (*dp) += 1;
    p += 1;
    (*dp) += 1;

    return EXIT_SUCCESS;
}
  
```

At this point in the code, what values are stored in `arr[]`?

- Vote at <http://PolEv.com/cse33320su>

A.	<code>{2, 3, 4}</code>	<code>arr[2]</code>	<code>4</code>
B.	<code>{3, 4, 5}</code>	<code>arr[1]</code>	<code>3</code>
C.	<code>{2, 6, 4}</code>	<code>arr[0]</code>	<code>2</code>
D.	<code>{2, 4, 5}</code>	<code>p</code>	<code>0x7fff...74</code>
E.	We're lost...	<code>dp</code>	<code>0x7fff...68</code>

32

32