Arrays & Algorithms

CS 150 – C++ Programming I
Lecture 11B

Array Review

- Array: built-in list of elements (homogenous)
  - `int a[5], b[] = {1, 2, 3};`
  - Name: address of first element, not a variable
  - Use subscript to access: `a[0]`
  - No range checking or exceptions
- Dereferencing: * and [] operators: any address
  - Combination of address and offset

`address[offset]` ⇔ `*(address + offset)`

  - `ptr + 2` is address expression (adds two elements to address)

Question

- Which displays the eighth element of a?

```
int a[15];

cout << a[8] << endl;          // A.
cout << a.at(7) << endl;       // B.
cout << a(7) << endl;          // C.
cout << a[7] << endl;          // D.
```

  - E. Runtime error because a is uninitialized

Question

- Which line has undefined output?

```
double speed[5]{...};

cout << speed[5] << endl;    // A.
cout << speed[0] << endl;    // B.
cout << speed[4] << endl;    // C.
cout << speed[1] << endl;    // D.
```

  - E. None of these

Question

```
int a[] = {1, 2, 3};
int b[] = {4, 5, 6};
a = b;
```

- What does the array `a` contain after this runs?
  - A. `{1, 2, 3}`
  - B. `{4, 5, 6}`
  - C. Undefined behavior
  - D. Syntax error - code will not compile

Question

- Which assigns to the first position in `letters`?

```
char letters[26];

letters[0] = "a";                  // A.
letters[1] = 'b';                  // B.
letters.front() = 'a';             // C.
letters = 'a';                     // D.
letters[0] = 'a';                  // E.
```

Question

- Assume `int dates[10];` What is the equivalent array notation for: `*(dates + 2);`
  - A. `dates[2]`
  - B. `dates[0] + 2`
  - C. `dates[2] + 2`
  - D. `&dates[2]`
  - E. `dates[0] + 4`

Question

- Assume `int dates[10];` What is the equivalent array notation for: `*dates + 2;`
  - A. `dates[2]`
  - B. `dates[0] + 2`
  - C. `dates[2] + 2`
  - D. `&dates[2]`
  - E. `dates[0] + 4`

Question

- Assume `int dates[10];` What is the equivalent array notation for: `(*dates + 2) + 2;`
  - A. `dates[2]`
  - B. `dates[0] + 2`
  - C. `dates[2] + 2`
  - D. `&dates[2]`
  - E. `dates[0] + 4`

Question

- Assume the following code. What prints?

```
int ar[] = {1, 2, 3, 4, 5};
int *p = ar + 2;
cout << *p++ << ",";
cout << *p << endl;
```

  - A. 2, 3
  - B. 3, 4
  - C. 4, 4
  - D. 4, 5

Question

- Assume the following code. What prints?

```
int ar[] = {1, 2, 3, 4, 5};
int *p = ar + 2;
cout << *++p << ",";
cout << *p << endl;
```

  - A. 2, 3
  - B. 3, 4
  - C. 4, 4
  - D. 4, 5

Question

- Assume the following code. What prints?

```
int ar[] = {1, 2, 3, 4, 5};
int *p = ar + 2;
cout << ++*p << ",";
cout << *p << endl;
```

  - A. 2, 3
  - B. 3, 4
  - C. 4, 4
  - D. 4, 5

Arrays, Functions & Pass-by-Address

- Arrays passed to functions by address not by reference
  - `int count(const int a[], size_t size);`
  - Parameter *a* is a pointer that is initialized with the address of first element in the array when called
  - Parameter may be written as `int a[]` or `int* a`
  - Unlike *vector*, you cannot pass entire array by value!
- The effect is similar to pass by reference
  - But don't use the & with an array parameter
- If array not changed, use `const` with the parameter (above)

Essential Algorithms

- Common algorithms are similar to *vector*
  - Must deal differently with the size problem
  - No range-checking, so extra care is required
- Some algorithm exercises similar to those we did earlier
  - Count for a particular condition: *counting.cpp*
  - Accumulate for a condition: *adding.cpp*
  - Find the extreme values: *extreme.cpp*

"Returning" Arrays

- You cannot return a local array (unlike *vector*)
  - Effect of pass-by-address is the same as pass-by-reference
  - Use that to create array output parameters
- Example: using a loop to fill an array with a value
  - ```
    void fill(int a[], size_t len, int val) {
        while (len) a[--len] = val;
    }
    ```
- Exercise: *output.cpp*
  - *tenRun* → Create runs of multiples of 10
    ```

Finish Up

- Before midnight, submit your In-Class exercise
- Before beginning of Lecture 12A
  - Complete the reading (links on Canvas)
  - Complete the reading exercises on Canvas
  - Take the quiz (2 tries, highest counted)
  - Complete H22, & H23 (due before 12A)
- Now – PE07 – Vectors & Structures
  - Put phones, laptops & backpacks at back or front of room
  - Sit at your assigned seats