Wrapping a static array

Written by Rachel J. Morris, last updated 1/30/17

Topics:

- Reviewing static arrays
- Pitfalls of static arrays
- Making a smarter array

Static arrays...

Arrays come in handy in our programs once we're storing lots of data of the same type.

We wouldn't want to have separate variables for each piece of data, because programming how we'd access that data would be tedious.

Static arrays…

Before we get into dynamic arrays in previous classes, we first work with static arrays.

<u>Static arrays…</u>

- **Must have a defined size at compile-time**
- **Cannot be resized while the program is running**
- **Have the risk of crashing the program if you go *out of bounds* of the array**

Static arrays...

Example program...

Static arrays...

1

```
int main()
{
    const int GAME_MAX = 5;
    int gameCount = 0;

    string games[GAME_MAX];

    bool done = false;
    while ( !done )
    {
        cout << "1. Add game" << endl
             << "2. Edit game" << endl
             << "3. Quit" << endl;
        cout << ">> ";
        int choice;
        cin >> choice;
        cin.ignore();
```

We have to keep track of both the array size AND amount of items entered in the array.

2

```
        switch( choice )
        {
            case 1: // Add game
            cout << "Game name: ";
            getline( cin, games[gameCount] );
            gameCount++;
            break;

            case 2: // Edit game
            cout << "Game index: ";
            int index;
            cin >> index;
            cin.ignore();

            cout << "Name update: ";
            getline( cin, games[index] );
            break;

            case 3: // Quit
            done = true;
            break;
        }
    }

    return 0;
}
```

Static arrays...

1

```
int main()
{
    const int GAME_MAX = 5;
    int gameCount = 0;

    string games[GAME_MAX];

    bool done = false;
    while ( !done )
    {
        cout << "1. Add game" << endl
             << "2. Edit game" << endl
             << "3. Quit" << endl;
        cout << ">> ";
        int choice;
        cin >> choice;
        cin.ignore();
```

2

```
        switch( choice )
        {
            case 1: // Add game
            cout << "Game name: ";
            getline( cin, games[gameCount] );
            gameCount++;
            break;

            case 2: // Edit game
            cout << "Game index: ";
            int index;
            cin >> index;
            cin.ignore();

            cout << "Name update: ";
            getline( cin, games[index] );
            break;

            case 3: // Quit
            done = true;
            break;
        }
    }

    return 0;
}
```

If we don't keep track of the # of items added, we could easily go out-of-bounds & crash the program.

Static arrays...

1

```
int main()
{
    const int GAME_MAX = 5;
    int gameCount = 0;

    string games[GAME_MAX];

    bool done = false;
    while ( !done )
    {
        cout << "1. Add game" << endl
             << "2. Edit game" << endl
             << "3. Quit" << endl;
        cout << ">> ";
        int choice;
        cin >> choice;
        cin.ignore();
```

2

```
        switch( choice )
        {
            case 1: // Add game
            cout << "Game name: ";
            getline( cin, games[gameCount] );
            gameCount++;
            break;

            case 2: // Edit game
            cout << "Game index: ";
            int index;
            cin >> index;
            cin.ignore();

            cout << "Name update: ";
            getline( cin, games[index] );
            break;

            case 3: // Quit
            done = true;
            break;
        }
    }

    return 0;
}
```

If we don't validate the user's input, they might give an invalid index, causing out-of-bounds access & program crash

Static arrays…

Having to write code to

- **Keep track of the array size *and* item count**
- **Validate user input**
- **Check whether the # of items added == array size**
- **Making sure nothing can access an invalid index**

Is extra work, and duplicate work, each time you have to implement a structure to store a list of data.

Static arrays…

Why not use classes to write a “smart array” that handles this for us ONCE, so we can just keep reusing that smart array over and over?

Static arrays…

Why not use classes to write a “smart array” that handles this for us ONCE, so we can just keep reusing that smart array over and over?

Now we’re starting to think about data structures

Smart array (static)

Let’s implement a smart array (using just a static array for now).

Additional functionality

Other functions to add to your smart array:

- **Insert** (at index, not just at the end)
- **Extend** (put one array at the end of another)
- **Remove** (at index)
- **Operator overloading:** (subscript [ ], =, ==)

Part of the challenge will be figuring out how this functionality should be implemented yourself.

Additional functionality

How to insert “Z” at position 2?
And how to test?

Index	0	1	2	3	4	5
Value	A	C	E	G	I	

Index	0	1	2	3	4	5
Value	A	C	Z	E	G	I