Can You Figure It Out?

- At a family reunion were the following people: one grandfather, one grandmother, two fathers, two mothers, four children, three grandchildren, one brother, two sisters, two sons, two daughters, one father-in-law, one mother-in-law, and one daughter-in-law. But not as many people attended as it sounds. How many were there, and who were they?

Arrays

Luke Sathrum – CSCI 20

Today's Class

- Introduction to Arrays
- Array Syntax
- Arrays and Memory

Introduction to Arrays

Introduction to Arrays

- Arrays are a way for us to store lots of information of the same type
- What are types?
  - Integers
  - Doubles
  - Strings
  - Classes
- Arrays behave like a list of variables with a uniform naming mechanism

Where might we use Arrays?

- Think of a program that reads in 5 test scores and performs some manipulations on these scores
- Maybe we want to compute the highest test score
- Then get the lowest test score
- How might we do this?

Array Example

- Here's our list of test scores
- 75, 90, 30, 85, 70
- Before we'd have to store each of these test scores in their own variable
  - `int test_score_1 = 75, test_score_2 = 90`, etc…
- Now we can put them together with one identifier

  `test_scores`

75	90	30	85	70

Array Example

- Let’s compute the highest test score
- We can also find the lowest test score

Array Syntax

Array Syntax

- To declare an array we do the following
  - `type name[number of variables];`
  - `int score[5];`
- The declaration is like declaring the following 5 variables of type `int`
  - `score[0], score[1], score[2], score[3], score[4]`
- Each of these is called an **element** of the array

Array Terms

- Index
  - The number in the brackets
    - `[0]` ← Index is 0
  - Each array's index starts at zero
- Declared Size
  - The number of elements in an array
  - If `score[7]` is the last element in an array then the size of the array is 8
  - It has elements 0 – 7 for a total of 8

Array Terms

- Base Type
  - The type that is shared by all elements in an array
    - `double score[5];`
    - `char score[5];`
    - `string score[5];`
- You can declare arrays inline with other variables
  - `int next, score[5], max;`

Array Terms

- Confusing the square brackets
  - In the declaration `[5]` means our array will have 5 elements
  - When we use the array `[4]` means the 5th element in the array

Array Terms

- The index in the brackets does not need to be an integer constant
  - You can have **`score[n + 1]`**
  - Because of this we use **`for`** loops often with arrays in C++

```
for (int i = 0; i < 5; i++)
        cout << score[i];
```

- This would output all the elements of the array

Sample Code

- Working with our first array
  - `score_array.cpp`

Initializing Arrays and Using Constants

Initializing Arrays

- An array can be initialized when it is declared

```
int children[3] = { 2, 12, 1 };
```

- This is equivalent to

```
int children[3];
children[0] = 2;
children[1] = 12;
children[2] = 1;
```

- If you do this you can omit the size of the array
  - `int children[] = { 2, 12, 1 };`

Defined Constants in Arrays

- **`children[]`** had a size of 3
- What if we want more? Less?
- Instead of declaring **`int children[3]`** let's use a constant

```
const int kNumberOfChildren = 5;
int children[kNumberOfChildren];
```

- Now we can handle changes to the size of our array at compile time

Defined Constants Notes

- You cannot use a variable for the array size, only a constant
- (Does Not Work)

```
int number;
cout << "Enter number of children:\n";
cin >> number;
int children[number];
```

Sample Code

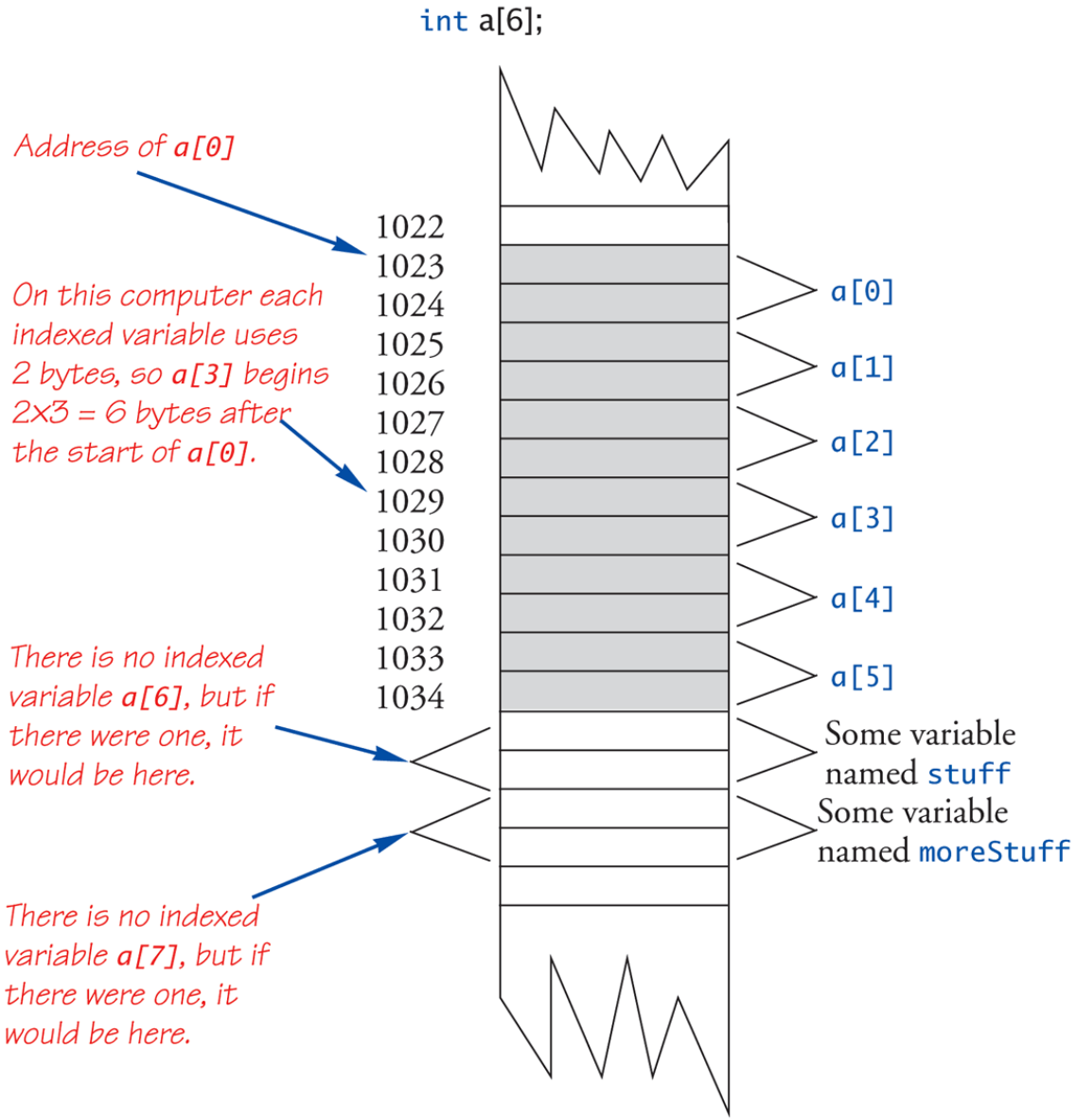
- Initializing Arrays and using Constants
  - `initialize_constant_array.cpp`

Arrays and Memory

Arrays in Memory

- Let's assume we have the following
  - `int a[6];`
- Each element is put in adjacent memory locations
- The computer only remembers the address of the first element
  - In our case **`a[0]`**
- It will compute the location of each other element as needed

Display 5.2 **An Array in Memory**



Array Index Out of Range

- The most common array error is attempting to reference a nonexistent array index
- For Example
  - `int a[4];`
  - What if we reference `a[4]`?
  - We call this an “index out of range” error

Array Index Out of Range

- This can be a problem in 2 different ways
  - Referencing **`a[4]`**

    `cout << a[4] << endl;`

    `a[4]` will have an unknown value
  - We can set **`a[4]`** to a value
    - **`a[4] = 100;`**
    - What happens here?

Array Index Out of Range

```
int a[3];
int stuff = 1;
int more_stuff = 2;
cout << a[4];
a[4] = 100;
```

1	a[0]	
2	|	
3	a[1]	
4	|	
5	a[2]	
6	|	
7	stuff	1
8	|	
9	more_stuff	2
10	|	
11		
12		

Review

- Arrays group values together of the same type
- Declared with square brackets
  - `int a[4];`
- Arrays start at index location _____?
- _____ loops are usually used with arrays
- Use curly braces to initialize arrays
- Use constants for the size of the array
- Arrays block out a chunk of memory
- Don’t go out of index!