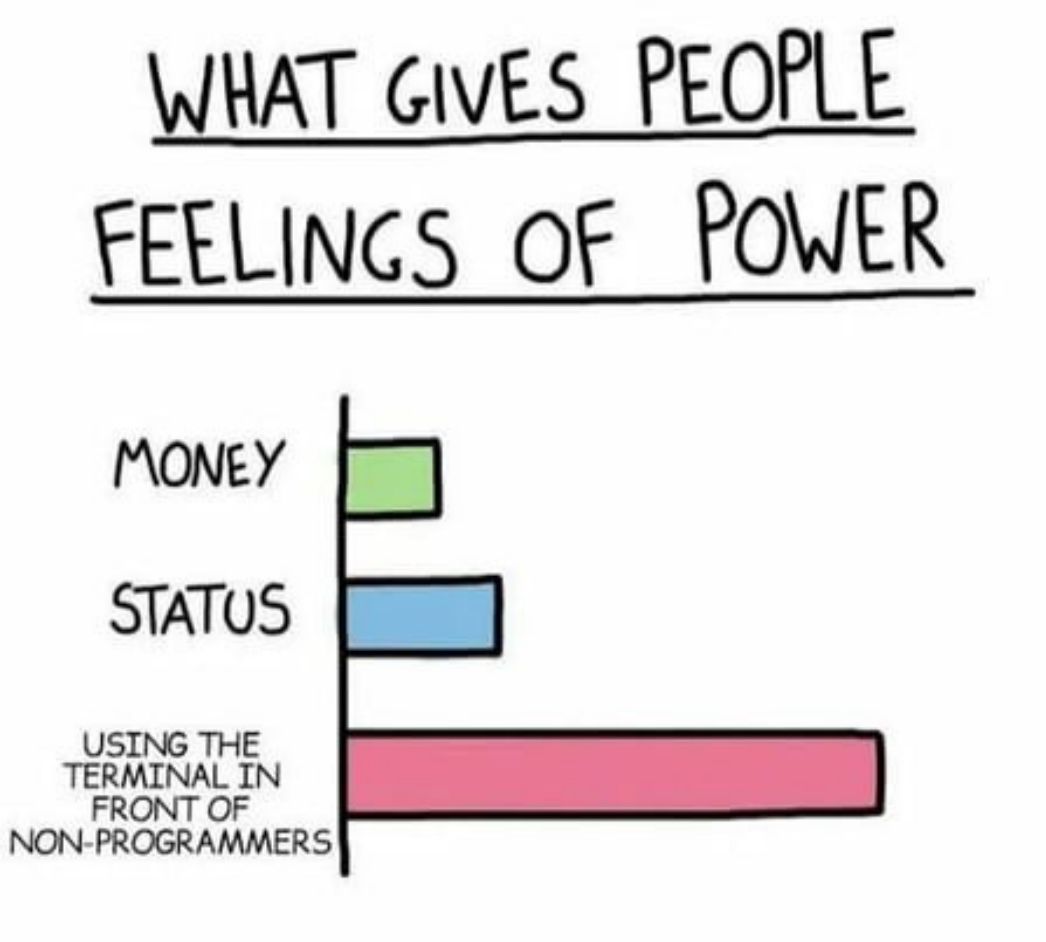

WHAT GIVES PEOPLE FEELINGS OF POWER
MONEY
STATUS
USING THE TERMINAL IN FRONT OF NON-PROGRAMMERS

CS1428
Foundation of Computer Science

Lecture 7: Arrays

Arrays

- Arrays are fixed-size collections of same-type values.
- Arrays are created using a variable declaration followed by [square brackets]
- We put the size of the array (the number of values) in the [square brackets].
- We reference the different values using a numerical index in the [square brackets].
- **The first index in the array is 0.**

A Very Simple Array

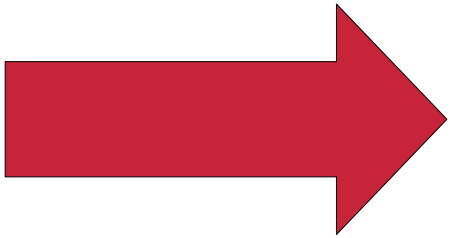
```
int main(){
    int a[3];
    a[0] = 1;
    a[1] = 2;
    a[2] = 3;
    cout << "The last value in the array is: " << a[2];
    return 0;
}
```

Simple Array Output

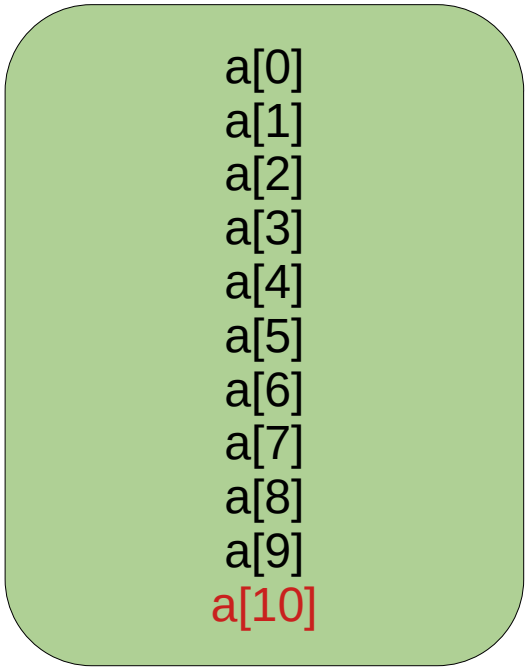
```
/home/gentry/Desktop/1428_testDir/junk1
The last value in the array is: 3
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

Arrays

int a[10];



Memory



Out of Bounds Errors

- Arrays have a size in memory equal to the size of one variable * the number of elements in the array.
  - So “**float x[10];**” allocates 32*10 bits or 40 bytes of memory.
- Indexing an array with a value that is less than 0 or greater than or equal to the size of the array access un-allocated memory. This is called referencing an “out of bounds”.
- Out of bounds references can crash a program or return junk values.

Out of Bounds Indexing

```
int main(){
    int a[3];
    a[0] = 1;
    a[1] = 2;
    a[2] = 3;
    cout << "The last value in the array is: " << a[3];
    return 0;
}
```

Out of Bounds Output

```
/home/gentry/Desktop/1428_testDir/junk1
The last value in the array is: -1373265920
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

```
/home/gentry/Desktop/1428_testDir/junk1
The last value in the array is: -1406522880
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

```
/home/gentry/Desktop/1428_testDir/junk1
The last value in the array is: 178629632
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

For an array a[SIZE]

The first index is a[0]

The last index is a[SIZE-1]

Using Global Constants for Size

```
const int SIZE = 10;

int main(){
    int list_of_numbers[SIZE];
    //do something with this array
    return 0;
}
```

Every Value in an Array has the Same Data Type

```
int main(){
    int list_of_numbers[SIZE];
    list_of_numbers[0] = 3;
    list_of_numbers[1] = 'c';
    list_of_numbers[2] = true;
    cout << list_of_numbers[0] << " " << list_of_numbers[1] << " "
        << list_of_numbers[2] << endl;
    return 0;
}
```

Array Output

```
/home/gentry/Desktop/1428_testDir/junk

3 99 1

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

For Loops + Arrays

- We often need to access every element (or many elements) in an array.
- A common method for this is to use a **for** loop with the loop variable (usually named i) being used as the index of the array.
- Remember: the first index is 0
- The condition must prevent out-of-bounds error.

For Lops and Arrays

```
int main()
{
    int a[10];
    for(int i = 0; i<10; i++)
        a[i] = i;
    for(int i = 9; i>=0; i--)
        cout << a[i] << " ";
    return 0;

}
```

Array Output

```
/home/gentry/Desktop/1428_testDir/junk
9 8 7 6 5 4 3 2 1 0
Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

2D Arrays

- Arrays can have more than one size value when the are declared.
- We can think of these two dimensional arrays as being tables with both rows and columns.
- The syntax for this is:
  - dataType name[row][column];
- The actual values are stored contiguously in memory grouped by row.

2D Arrays

int a[3][3];

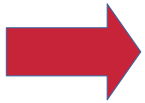
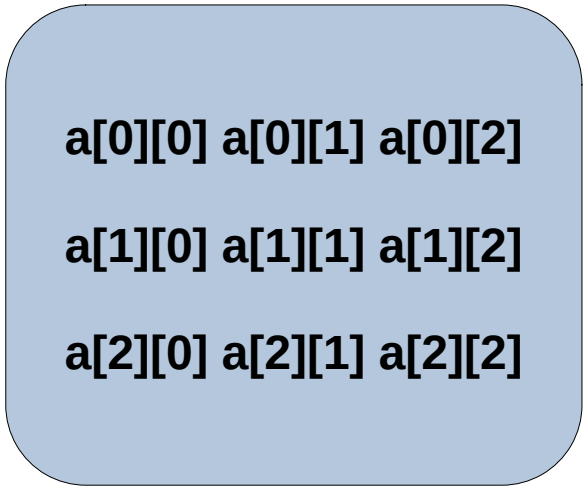
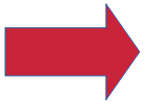
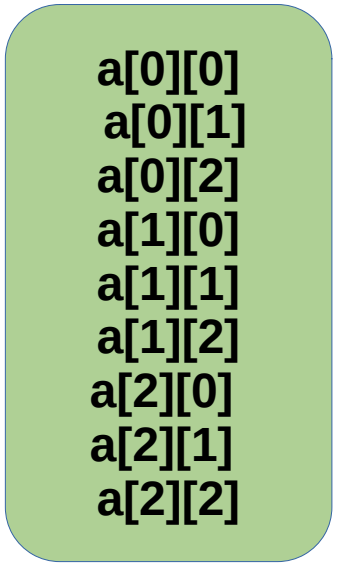


Table Abstraction





Actual Memory



Accessing 2D Arrays

```
int main()
{
    int a[3][3];
    a[0][0] = 1;
    a[0][1] = 2;
    a[0][2] = 3;
    cout << "First row of a: " << a[0][0] << a[0][1] << a[0][2] << endl;
    return 0;

}
```

Array Output

```
/home/gentry/Desktop/1428_testDir/junk
First row of a: 123

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

For Loops + 2D arrays

```
int main()
{
    int a[10][10];
    for(int i = 0; i < 10; i++){
        for(int j = 0; j<10; j++){
            a[i][j] = i+j;
        }
    }

    for(int i = 0; i < 10; i++){
        for(int j = 0; j<10; j++){
            cout << a[i][j] << "\t";
        }
        cout << endl;
    }
    return 0;

}
```

Array Output

```
/home/gentry/Desktop/1428_testDir/junk

0       1       2       3       4       5       6       7       8       9
1       2       3       4       5       6       7       8       9       10
2       3       4       5       6       7       8       9       10      11
3       4       5       6       7       8       9       10      11      12
4       5       6       7       8       9       10      11      12      13
5       6       7       8       9       10      11      12      13      14
6       7       8       9       10      11      12      13      14      15
7       8       9       10      11      12      13      14      15      16
8       9       10      11      12      13      14      15      16      17
9       10      11      12      13      14      15      16      17      18

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

Initializing Arrays

- We can initialize arrays with comma separated list of values inside of {curly brackets}.
- This can only be done on the same line as the array declaration.
- You do not have to explicitly give the size of the array if you are using an initialization list.

Initializing Arrays

```
int main()
{
    char a[] = {'B', 'o', 'k', 'o', '\0'};
    cout << a << endl;
    return 0;
}
```

Array Output

```
/home/gentry/Desktop/1428_testDir/junk

Boko

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

‘C’ Strings

- String as a data type is a new-ish addition to C++.
- The C programming language used arrays of characters instead of strings.
- Every C-string has to be terminated with a special character called the “null terminator”, written like this ‘\0’
- This makes the size of C-strings one greater than the number of characters in them.
- Some older functions prefer to use C-strings rather than strings.

2D Initialization Lists

```
int main()
{
    //Must explicitly size one dimension
    int a[][3] = {{1, 2, 3}, {2, 3, 4}, {3, 4, 5}};
    for(int i = 0; i<3; i++){
        for(int j = 0; j< 3; j++){
            cout << a[i][j] << '\t';
        }
        cout << endl;
    }
}
```

Array Output

```
/home/gentry/Desktop/1428_testDir/junk

1       2       3
2       3       4
3       4       5

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

Multi-dimensional Arrays

- We don’t have to stop at 2-dimensional arrays, we can continue to add an arbitrary number of dimensions.
- The syntax for a 4-D array looks like this:
  - int a[SIZE][SIZE][SIZE][SIZE];
- The usefulness of higher dimensional arrays diminishes as the get harder to visualize and process.
- It is often better to use more, lower dimensional arrays than it is to use on high dimensional array.

Other Containers in C++

- An array is only one method for storing multiple values in a single named “variable”.
- C++ also provides:
  - Vector
  - List
  - Dequeue
  - Many others
- We won’t be looking at these any further in this class, but they can be helpful.

The first valid index in an array is 0!!!