CISC 1600/1610
Computer Science I

Arrays

Professor Daniel Leeds
dleeds@fordham.edu
JMH 328A

Data types

Single pieces of information
• one integer – int
• one symbol – char
• one truth value – bool

Multiple pieces of information
• group of symbols – string
• group of anything – array

An array is a list containing
• a fixed number of entries AND
• entries of all the same type

int a[5]; - declares an array of 5 ints
float c[8]; - declares an array of 8 floats

Array syntax

• Declaring an array:
  char grades[4];

• Initializing an array:

• Accessing an array element:
  cout << grades[2];

Zero-indexing

• An array with n elements is accessed with indices 0 through n-1

  int dailyTemps[4] - accesses fifth element of the dailyTemps array

Memory allocation

Declaration of array with n elements

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>04902340</td>
<td>12</td>
</tr>
<tr>
<td>04902348</td>
<td></td>
</tr>
<tr>
<td>04902356</td>
<td></td>
</tr>
<tr>
<td>04902364</td>
<td></td>
</tr>
<tr>
<td>04902372</td>
<td></td>
</tr>
<tr>
<td>04902380</td>
<td></td>
</tr>
<tr>
<td>04902388</td>
<td></td>
</tr>
<tr>
<td>04902396</td>
<td></td>
</tr>
<tr>
<td>04902404</td>
<td></td>
</tr>
<tr>
<td>04902412</td>
<td></td>
</tr>
<tr>
<td>04902420</td>
<td></td>
</tr>
<tr>
<td>04902428</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>04902340</td>
<td>12</td>
</tr>
<tr>
<td>04902348</td>
<td>89</td>
</tr>
<tr>
<td>04902356</td>
<td>543</td>
</tr>
<tr>
<td>04902364</td>
<td></td>
</tr>
<tr>
<td>04902372</td>
<td></td>
</tr>
<tr>
<td>04902380</td>
<td></td>
</tr>
<tr>
<td>04902388</td>
<td></td>
</tr>
<tr>
<td>04902396</td>
<td></td>
</tr>
<tr>
<td>04902404</td>
<td></td>
</tr>
<tr>
<td>04902412</td>
<td></td>
</tr>
<tr>
<td>04902420</td>
<td></td>
</tr>
<tr>
<td>04902428</td>
<td></td>
</tr>
</tbody>
</table>
### Declaration

Array must* be declared with constant number of entries

```cpp
const int gradeSize=26;
char grades[gradeSize];
float heights[26];
```

### Initialization

- Entries of array can be initialized with bracketed list
- Un-filled slots will default to zero after initialization

```cpp
float heights[26]={5.5, 4.9, 6, 4.5};
cout << heights[1] << " " << heights[6]; // Outputs: 4.9 0
```

### Arrays and loops

For loops are well-structured to handle arrays

```cpp
const int gradeSize=26;
char grades[gradeSize];
for(int i=0; i<gradeSize; i++) {
    cout << grades[i] << endl;
}
```

### What does this code do?

```cpp
int a[5]={1,3,6,4,2};
cout << a[3] << endl;
int i=1;
while(i<4) {
    cout << a[i+1]-a[i] << endl;
    i++;
}
```

### What does this code do?

```cpp
int a[5]={1,3,6,4,2}
int b[5], size_b=0;
int i=0;
while(i<4) {
    if (a[i]>3) {
        b[size_b]=a[i];
        size_b++;
    }
    i++;
}
```

### Out-of-range indexing

- An index value not allowed by array declaration is “out of range”
  ```cpp
  char a[10];
cin >> a[10]; // out of range!
  ```
- Out-of-range index produces no compiler error, but can cause serious program problems
  - Reading/writing incorrect spots in memory
Out-of-range indexing

```cpp
int scores[4]={1,2}, idNum;
idNum=34253;
scores[5]=12;
cout << idNum;
```

Array elements in functions

```cpp
• Array element accepted as normal function argument

```cpp
int my_function(int n);
int a[10], b;
Then can execute:
b=my_function(a[2]);
b=my_function(a[5]);
```

Arrays in functions

We can pass full array to a function

- Function declaration
  ```cpp
  void printList(int list[], int size);
  ```
- Call
  ```cpp
  int list[5], size=5;
  printList(list, size);
  ```

Roughly “pass by reference”

- By default, elements of input array can be changed by function
  ```cpp
  void getList(char a[], int size);
  // Precondition: Receives blank list
  // of chars and size of list
  // Postcondition: list of chars is
  // filled by user
  ```

Roughly “pass by reference”

- Function will see variable type and memory location of first element
- Useful to include formal parameter reporting array size

```cpp
void getList(char a[], int size);
// Precondition: Receives blank list
// of chars and size of list
// Postcondition: list of chars is
// filled by user
```
Programming with arrays

• Search – is number x in my array?
• Sort – arrange numbers from small to large

Sorting method

Start with:
• Unsorted list of numbers U
• Empty list E
Method:
• Find smallest number in U
• Place smallest number in E (E[0]=smallest(U);)
• Find second-smallest number in U
• Place second-smallest number in E
• ...keep going

Functions: const arrays

• Can insist array values remain unchanged:
  • Function declaration
    void showAll(const int list[], int size);
  • Call
    int list[5], size=5;
    showAll(list, size);

More on const arrays

• If formal parameter is const array, cannot input to another function as non-const
  void showAll(const int list[], int size);
  void getAll(int list[], int size);
  ...
  void showAll(const int list[], int size){
    getAll(list, size); // ERROR!
    ...
    // Display list elements
    return;
  }

Multi-dimensional arrays

• Storing a table of data
  const int numStudents=5, numTests=3;
  char grades[numStudents][numTests];
  grades[2][0] = 'A';
  grades[3][0] = 'B';

  grades
```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

“Array of arrays”

char grades[5][3] treated as array with 5 entries – each “entry” is a 3-element char array

```
grades[0] A 0
  B 1
  A 2
grades[1] C 0
  C 1
  B 2
grades[2] A 0
  B 1
```
Passing multi-dimensional arrays

void print_list(const char list[][3],
               int num_rows);

Size of inner array must be specified