

## Dynamic Arrays

CS 1044  
Summer I 2004

---

---

---

---

---

---

---

---

## Array Definition Revisited

**<type> <varname>[<size>;**

Example: **int table[20];**

**<size>** must be constant

---

---

---

---

---

---

---

---

## Possible Use for a Variable-Sized Array

```
int numEntries;  
cin >> numEntries;  
double values[numEntries];  
    // above definition is not allowed  
    // because numEntries is a variable  
for ( int i = 0; i < numEntries; i++ )  
    cin >> values[i];
```

---

---

---

---

---

---

---

---

## Solution: Dynamically Allocated Arrays

```
cin >> numEntries;
double *values = new double[numEntries];
// use values just like an array, from here on
for ( int i = 0; i < numEntries; i++ )
    cin >> values[i];
...
delete [] values; // frees up allocated memory
```

---

---

---

---

---

---

---

---

## Syntax

- Declare pointer to <type>  
– <type> \*<var>; e.g., int \*scores;
- Allocate memory  
– <var> = new <type>[<int-expr>;  
e.g., scores = new int[size];
- Free memory  
– delete [] var; e.g., delete [] scores;

---

---

---

---

---

---

---

---

## Two-Dimensions

```
int **table;
table = new int*[numRows];
for(i = 0; i < numRows; i++)
    table[i] = new int[numColumns];
// use table like regular 2-D array...
for(i = 0; i < numRows; i++)
    delete [] table[i];
delete [] table;
```

---

---

---

---

---

---

---

---

# Introduction to OOP and C++ Classes

CS 1044  
Summer I 2004

---

---

---

---

---

---

---

---

## Object

**Definition:** a thing that has identity, state, and behavior

- identity: a distinguished instance of a **class**
- state: collection of values for its **variables**
- behavior: capability to execute its own **functions or methods**

\* variables and methods are defined in a class

---

---

---

---

---

---

---

---

## Class

**Definition:** a collection of attributes (variables) and methods (functions) that carry out operations on those variables

variables and functions define the contents and capabilities of the instances (objects) of the class

---

---

---

---

---

---

---

---

## C++ Class Example

```
class bankaccount {  
  double balance; ← attribute  
public:  
  bankaccount() {  
    balance = 0; ← initialize attributes in  
                  constructor (special method that  
                  has the same name as the class)  
  }  
  void deposit( double amount ) {  
    balance = balance + amount; ← methods  
  }  
  double getbalance() {  
    return balance; ← methods  
  }  
};
```

---

---

---

---

---

---

---

---

## Using Classes

```
bankaccount myacct;  
// define a variable (or object)  
// just like with other types  
  
myacct.deposit( 100.00 );  
cout << myacct.getbalance() << endl;  
// method invocation just like functions
```

---

---

---

---

---

---

---

---

## Some Familiar Classes

- ifstream and ofstream  
(defined in <fstream>)
  - Methods: fail(), ignore()
- string (defined in <string>)
  - Methods: length(), and the concatenation operator (+)

---

---

---

---

---

---

---

---