Arrays

Chapter 8

page 471

Arrays (8.1)

- One variable that can store a *group of values of the same type*

- Storing a number of related values
  - all grades for one student
  - all temperatures for one month
  - hours worked for each day

Arrays

```
int age = 42;


int ages[3];

// datatype variable_name [ size ];

const int CLASSSIZE = 24;
string names[CLASSSIZE];
```

The size of the array must be a *literal* or a `const int`.

Using arrays (8.2)

- The first element in the array is the **0th** element!
- The ***index*** is an `int`

```
int y, x = 3;
int years[10];

years[0] = 2;
years[x] = 4;
y = years[0] + 9;
```

Practice

- Declare an array to hold the height, in inches, of six trees.
- Set the height of the trees as:
    - 32 inches
    - 45 inches
    - 99 inches
    - 120 inches
    - 500 inches
    - 600 inches

Practice (8.3)

- Write a snippet of code to print to the screen every value in this array:

```
const int ARRAYSIZE = 4;
int vals[ARRAYSIZE];
```

- Print the sum and average

Practice

- Read 20 exam scores from a file and print them in reverse order
- Ask the user for an exam number (0-19) and print that exam score to the screen
- Ask the user for an exam number and add 2 bonus points to that exam score.
- Find the max score in the array

Out of bounds (p 479)

- C++ does *not* check to make sure the ***index*** falls within the array
  - no *bounds checking*
  - this will cause unpredictable results!

Initialization (8.4)

- What is the equivalent of:

```
int value = 2; // initialize the variable

int tests[2] =
string names[3] =
```

- Initialize just a few values:

```
int value[4] =
```

Implicit array sizing (p 486)

- Set the size of the array by initializing it
- You *must* either specify a size or initialize the array

```
string names[] =


char letters[] =
```

Arrays and Functions

- Pass an array as an argument

```
void printArray(int arr[], int arraySize)
{

}

int main()
{
  const int MAXPEOPLE = 100;
  int ages[MAXPEOPLE];

  printArray(ages, MAXPEOPLE);
```

Practice

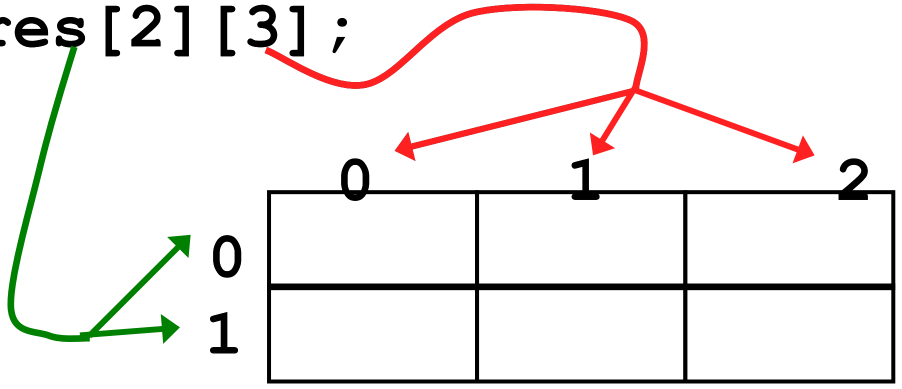
- Write a function that will accept an integer array and a size for that array, and return the sum of all the elements in the array

```
    int sumArray(int array[], int size);
```

Two dimensional arrays (8.9)

- A grid of data!

```
int testScores[2][3];
```



```
testScores[0][0] = 99;
testScores[0][1] = 80;
testScores[0][2] = 88;
testScores[1][0] = 89;
testScores[1][1] = 77;
testScores[1][2] = 85;
```

Why use 2D arrays?

- Hold the scores for each student in one array.

```
const int BOB = 0;
const int ALICE = 1;
const int MIDTERM1 = 0;
const int MIDTERM2 = 1;
const int FINAL = 2;
int testScores[2][3] = { {0, 0, 0},
                         {0, 0, 0} };
testScores[BOB][MIDTERM1] = 99;
testScores[ALICE][FINAL] = 85;
```

- Which values are we setting above?
- How do we set Alice's Midterm2 score?
- What is stored in **`testScores[0][1]`** ?

Practice

- Use a two dimensional array to store the scores of 8 Pacific Volleyball games. Store the opponent names in a separate one dimensional array. Read these values from PV.txt. Pacific's score is listed first

```
Concordia 3 2
Schreiner 3 0
Wartburg 3 2
Iowa 3 2
LaVerne 3 2
UCSC 2 3
CalLutheran 0 3
Pomona 2 3
```

- Print the name of the first team that Pacific beat
- Print the name of the last team that Pacific beat
- Print the name of the first team that beat Pacific

Practice

- Using the array below, calculate:
    - the average score on each assignment
    - the average score for each student
    - assume the array already contains data

```
const int NUMOFSTUDENTS = 24;
const int NUMOFASSIGNMENTS = 6;

int testScores[NUMOFSTUDENTS][NUMOFASSIGNMENTS];
```

N-Dimensional Arrays (8.10)

```
// cost of seats in a theatre
//
// 4 sections, each section has
// 20 rows with 30 seats each.

double seats[4][20][30];

seats[0][0][0] = 100.00;
seats[2][0][0] = seats[1][0][0] / 2;
seats[3][19][25] = 10.00;

// we can have as many dimensions as
// necessary in an array
```