Chapter 5

Arrays

Chapter 5 Topics

Declaring and Using an One-Dimensional Array

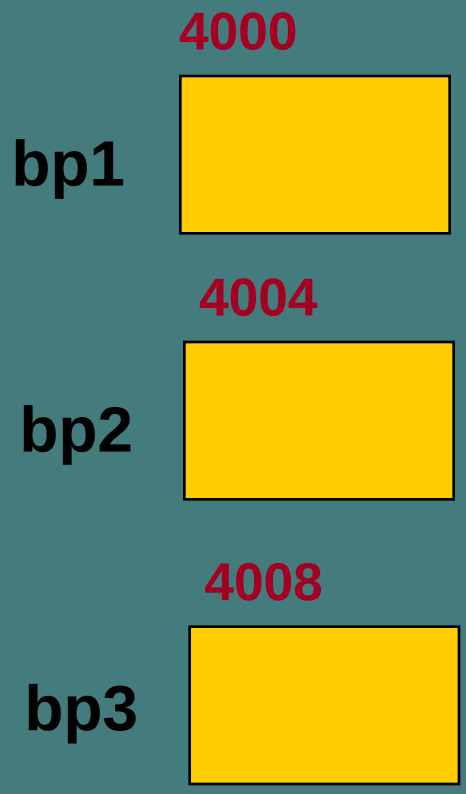
- Array Declaration
- Accessing Array Elements
- Passing Arrays as Arguments to Functions

Declaring and Using a Two-Dimensional Array

- Array Declaration
- Accessing Array Elements
- Storage of Array
- Arrays as Parameters

Declare variables to store and total 3 blood pressures

```
int  bp1,  bp2,  bp3;
int  total;
```




```
cin >> bp1 >>  bp2 >>  bp3;
total =   bp1 +  bp2 +  bp3;
```

What to do if you wanted to store and total 1000 blood pressures

One-Dimensional Array Definition

An array is a structured collection of components (called array elements), all of the same data type, given a single name, and stored in adjacent memory locations.

Declaration of an Array

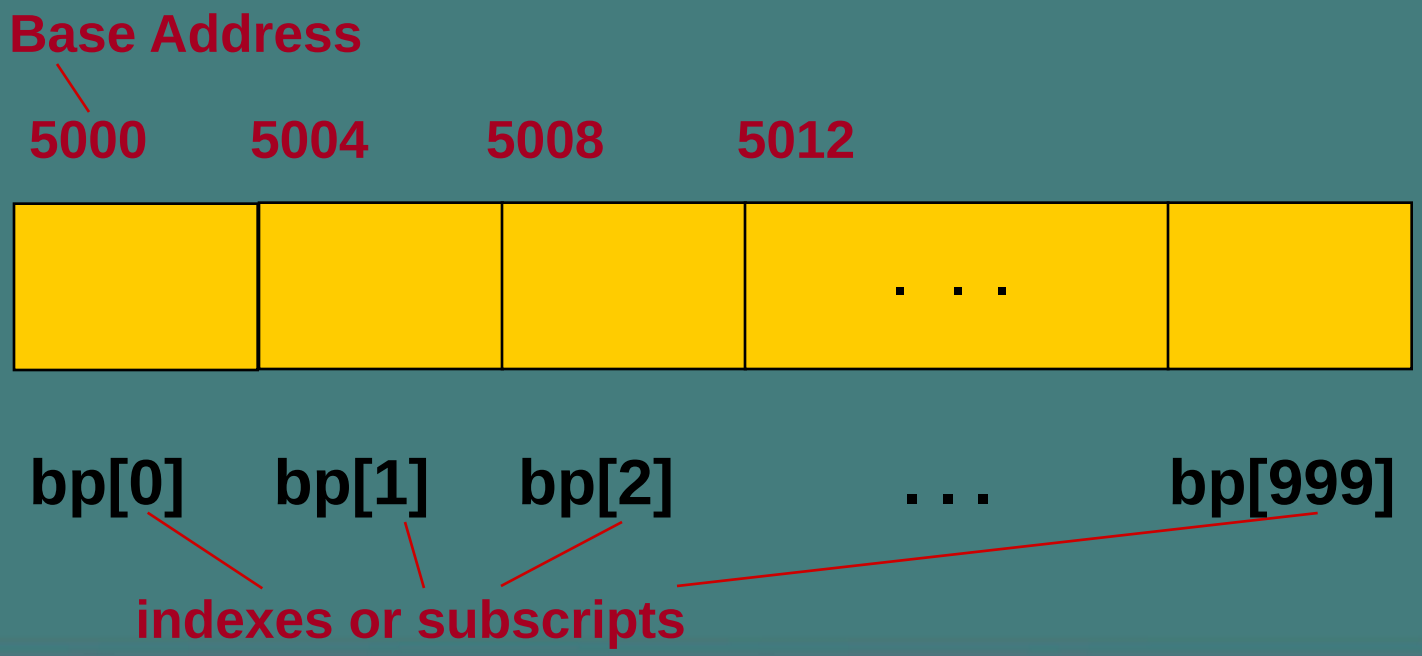
SYNTAX

```
DataType  ArrayName [ConstIntExpression];
```

- the index is also called the subscript
- the base address of an array is its beginning address in memory

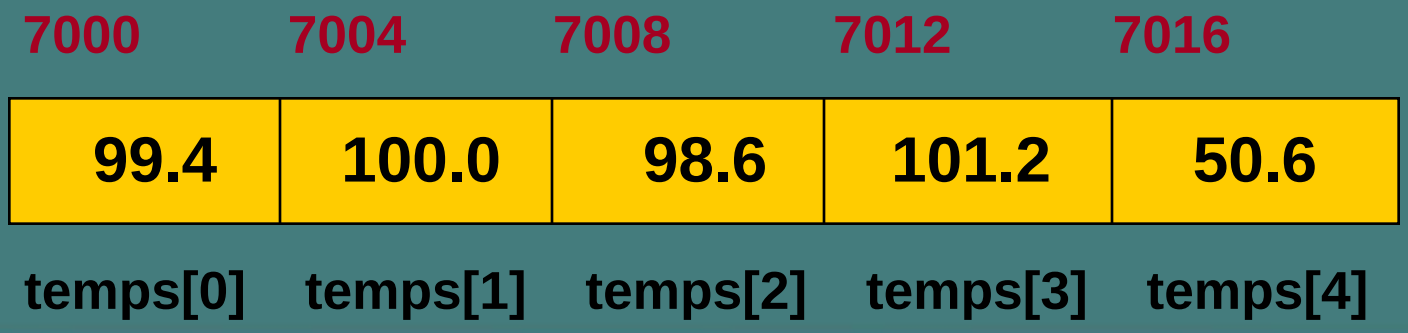
What to do if you wanted to store and total 1000 blood pressures?

```
int  bp[ 1000 ] ;  // declares an array of 1000 int values
```



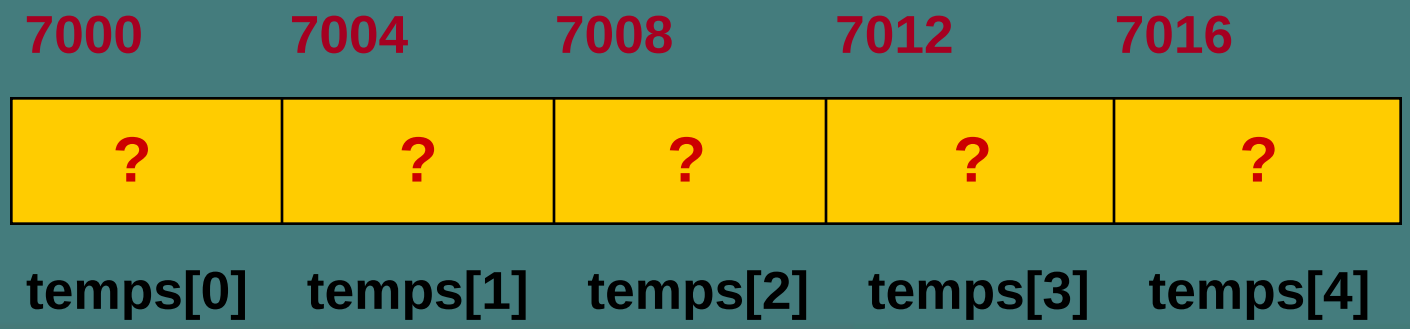
Assigning Values to Individual Array Elements

```
float  temps[ 5 ] ;          // allocates memory for array
int    m = 4 ;
temps[ 2 ] = 98.6 ;
temps[ 3 ] = 101.2 ;
temps[ 0 ] = 99.4 ;
temps[ m ] = temps[ 3 ] / 2.0 ;
temps[ 1 ] = temps[ 3 ] - 1.2 ;
```



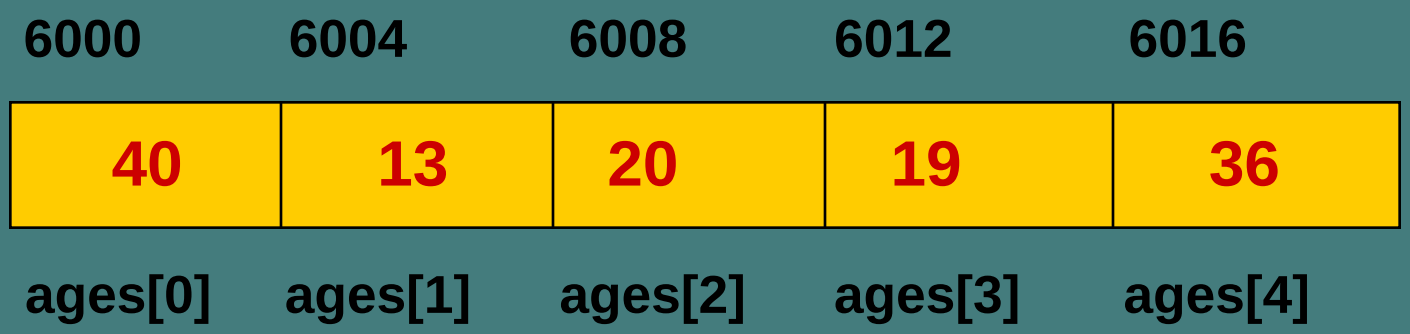
What values are assigned?

```
float  temps[ 5 ] ;        // allocates memory for array
int    m ;

for (m = 0; m < 5; m++)
{
    temps[ m ] = 100.0 + m * 0.2  ;
}
```



Initializing in a Declaration

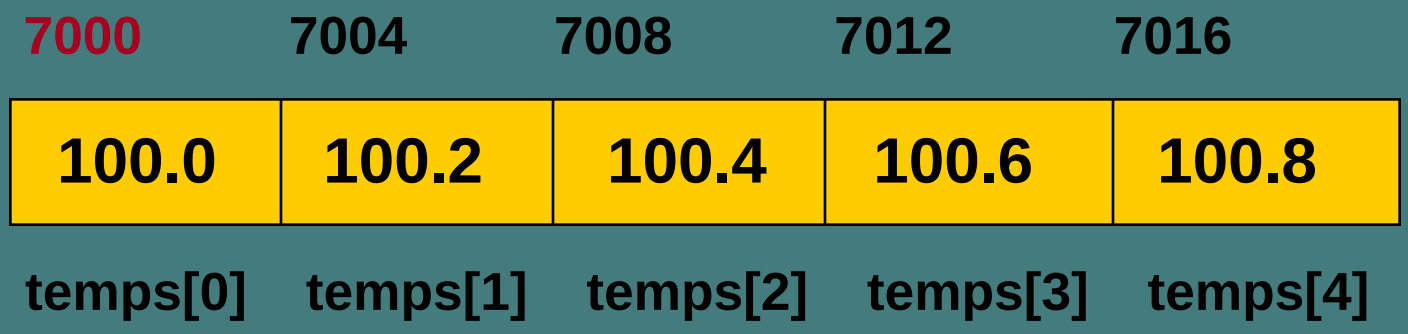
```
int   ages[ 5 ]  =  { 40, 13, 20, 19, 36 } ;

for  ( int m = 0; m < 5; m++ )
{
    cout << “ages[ ”<< m<< “ ]= ” <<ages[ m ]<<endl ;
}
```



A Closer Look at the Compiler

```
float  temps[5];   // this declaration allocates memory
```

To the compiler, the value of the identifier **`temps`** alone is the base address of the array. We say **`temps`** is a pointer (because its value is an address). It "points" to a memory location.



Passing Arrays as Arguments to Functions

- in C++, arrays are *always* passed by reference
- whenever an array is passed as an argument, its base address is sent to the called function

Example with Array Parameters

```
#include <iomanip>
#include <iostream>

void  Obtain ( int [ ], int ) ;                          // prototypes here
void  FindWarmest ( const  int[ ],  int , int & ) ;
void  FindAverage  ( const  int[ ],  int , int & ) ;
void  Print ( const  int [ ], int ) ;


using  namespace  std ;

int main (   )
{
    int    temp[31] ;        // array to hold up to 31 temperatures
    int    numDays ;
    int    average ;
    int    hottest ;
    int    m ;
```

Example continued

```
    cout  <<  "How many daily temperatures? " ;
    cin  >>  numDays ;

    Obtain( temp, numDays ) ;    // passes value of numDays and
                                 // address of array temp to function

    cout  <<  numDays  <<  " temperatures"  << endl ;
    Print ( temp, numDays ) ;

    FindAverage ( temp, numDays, average ) ;
    FindWarmest ( temp, numDays, hottest ) ;

    cout  <<  endl  <<  "Average was:  " << average  << endl ;
    cout  <<  "Highest was:  "  << hottest  << endl ;


    return 0 ;
}
```

Chapter 5 Topics

- Declaring and Using an One-Dimensional Array
  - Array Declaration
  - Accessing Array Elements
  - Passing Arrays as Arguments to Functions
- Declaring and Using a Two-Dimensional Array
  - Array Declaration
  - Accessing Array Elements
  - Storage of Array
  - Arrays as Parameters

Two-Dimensional Array

is a collection of components, all of the same type, structured in two dimensions(referred to as rows and columns).

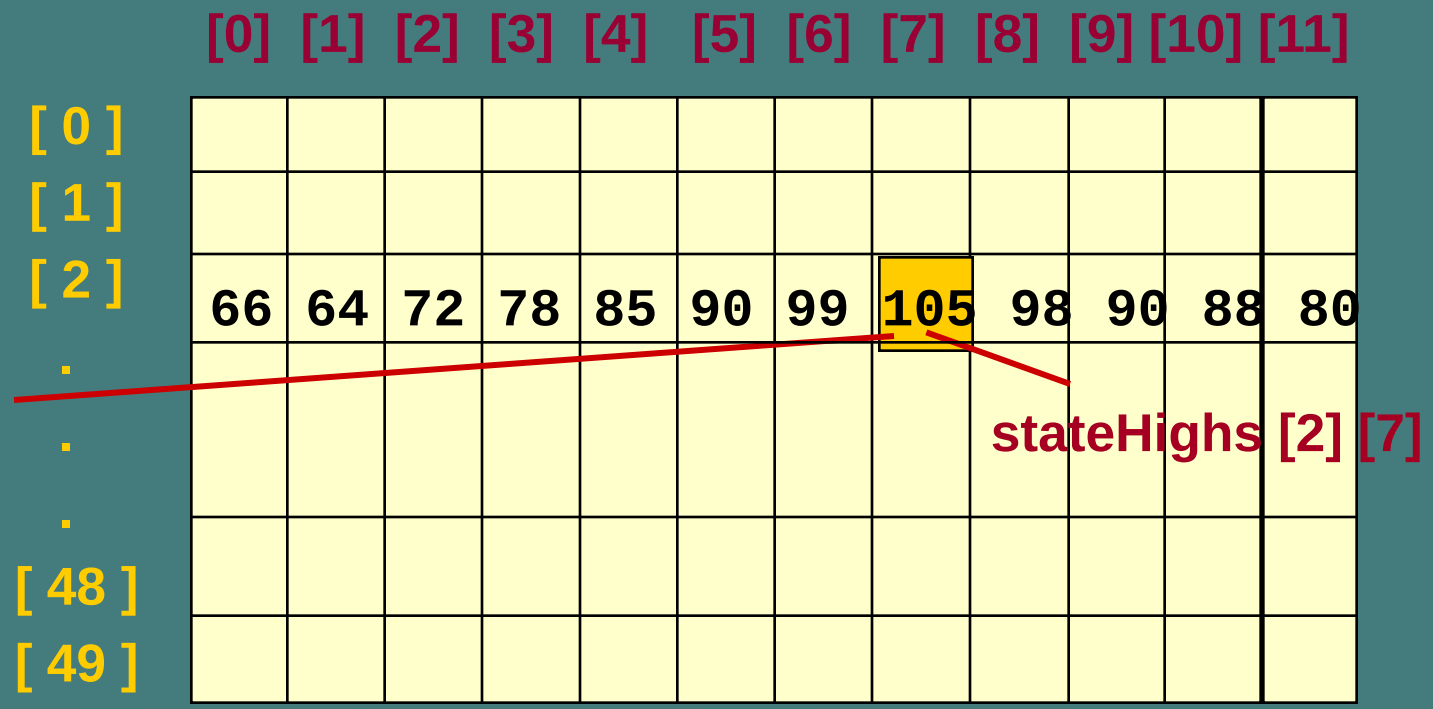
Individual components are accessed by a pair of indexes representing the component's position in each dimension.

SYNTAX FOR DECLARATION

```
DataType   ArrayName [ConstIntExpr]  [ConstIntExpr] ;
```

EXAMPLE -- To keep monthly high temperatures for all 50 states in one array.

```
const int NUM_STATES   = 50 ;
const int NUM_MONTHS = 12 ;
int stateHighs [ NUM_STATES ] [ NUM_MONTHS ] ;
```



Finding the average high temperature for Arizona

```
int  total  = 0 ;
int  month ;
int  average ;
for ( month = 0 ; month < NUM_MONTHS ; month ++ )
     total = total + stateHighs [ 2 ] [ month ] ;
average  = int ( total / 12.0  + 0.5 ) ;
```

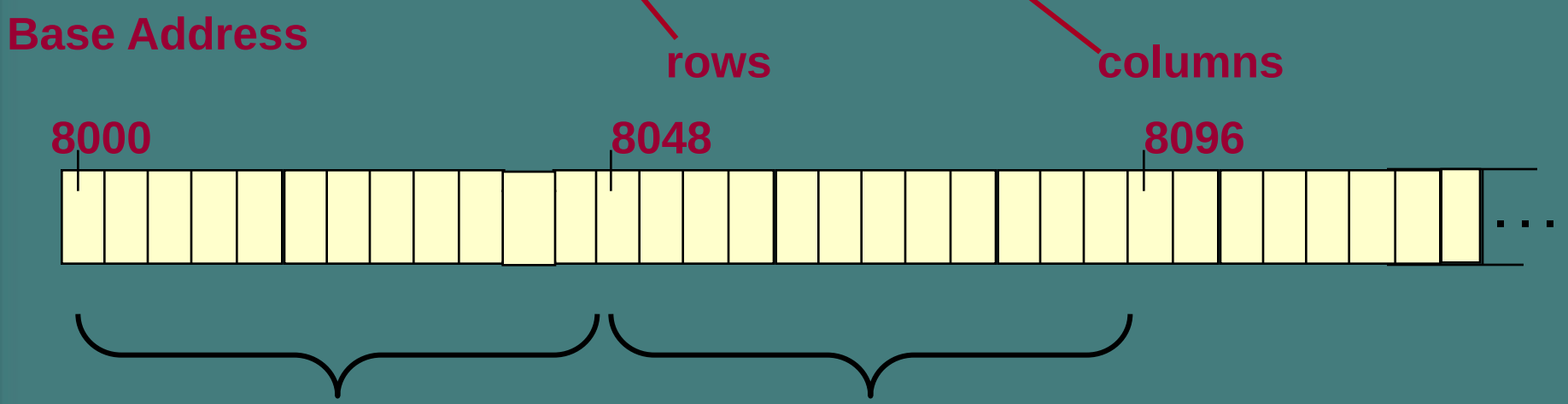
average

85

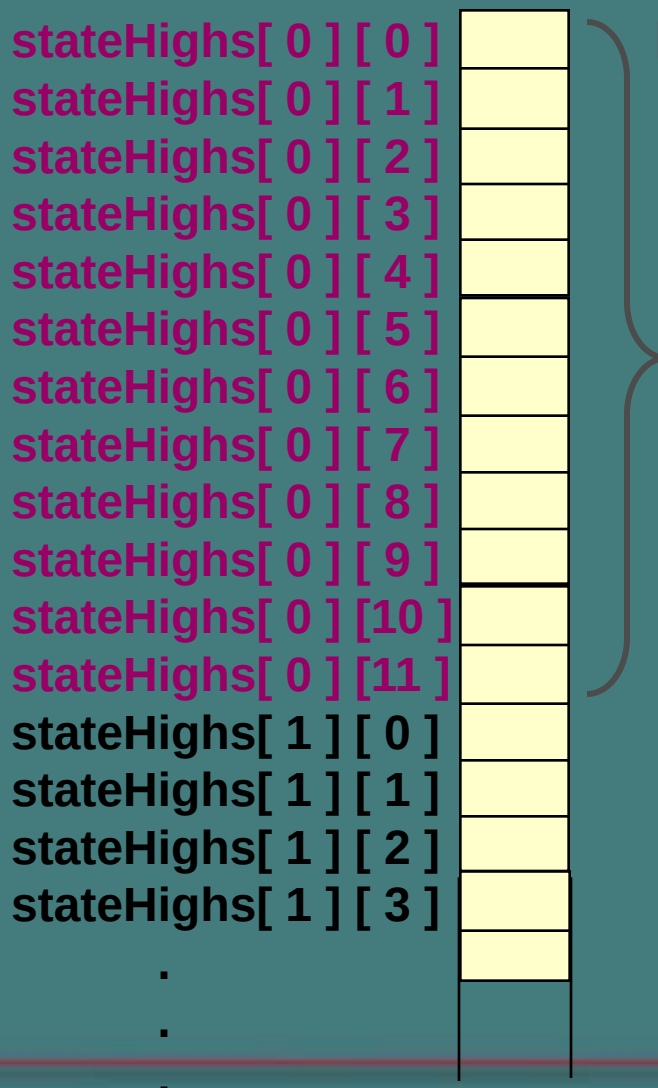
STORAGE

- In memory, C++ stores arrays in row order. The first row is followed by the second row, etc.

```
const  int  NUM_STATES   =  50 ;
const  int  NUM_MONTHS  =  12 ;
int  stateHighs [ NUM_STATES ] [ NUM_MONTHS ] ;
```



Viewed another way . . .



Base Address 8000

To locate an element such as stateHighs [2] [7] the compiler needs to know that there are 12 columns in this two-dimensional array.

At what address will stateHighs [2] [7] be found?

Assume 4 bytes for type int.

Arrays as Parameters

- just as with a one-dimensional array, when a two- (or higher) dimensional array is passed as an argument, the base address of the caller's array is sent to the function
- the size of all dimensions except the first must be included in the function heading and prototype
- the sizes of those dimensions in the function's parameter list must be exactly the same as declared for the caller's array