Introduction to C++ Programming: Lecture 4

Presented by

DR. MOHAMMED ATTIA MAHMOUD

-PhD, Fayoum University, Egypt and Antwerp University, Belgium.

-Researcher in ENHEP, ASRT, Fayoum Uni, and BUE.

-FSQ Gen-Contact, CMS experiment, CERN, Geneva, Switzerland.

Arrays

- An array is a collection of values that have the same data type, e.g.
  - A collection of int data values or
  - A collection of bool data values
- We refer to all stored values in an array by its name
- If we would like to access a particular value stored in an array, we specify its index (i.e. its position relative to the first array value)
  - The first array index is always 0
  - The second value is stored in index 1
  - Etc.

Examples Using Arrays

- ***Initializing arrays***
  - For loop
    - Set each element
  - Initializer list
    - Specify each element when array declared

    ```
    int n[ 5 ] = { 1, 2, 3, 4, 5 };
    ```
    - If not enough initializers, rightmost elements 0
    - If too many syntax error
  - To set every element to same value

    ```
    int n[ 5 ] = { 0 };
    ```
  - If array size omitted, initializers determine size

    ```
    int n[] = { 1, 2, 3, 4, 5 };
    ```
    - 5 initializers, therefore 5 element array

```
// Fig. 4.3: fig04_03.cpp
// Initializing an array.
#include <iostream>
    using std::cout;
using std::endl;

#include <iomanip>

using std::setw;

int main()
{
  int n[ 10 ];  // n is an array of 10 integers

  // initialize elements of array n to 0
  for ( int i = 0; i < 10; i++ )
    n[ i ] = 0;   // set element at location i to 0

  cout << "Element" << setw( 13 ) << "Value" << endl;

  // output contents of array n in tabular format
  for ( int j = 0; j < 10; j++ )
    cout << setw( 7 ) << j << setw( 13 ) << n[ j ] << endl;
   return 0;  // indicates successful termination
} // end main
```

Declare a 10-element array of integers.

Initialize array to `0` using a for loop. Note that the array has elements `n[0]` to `n[9]`.

```
Element        Value
      0            0
      1            0
      2            0
      3            0
      4            0
      5            0
      6            0
      7            0
      8            0
      9            0
```

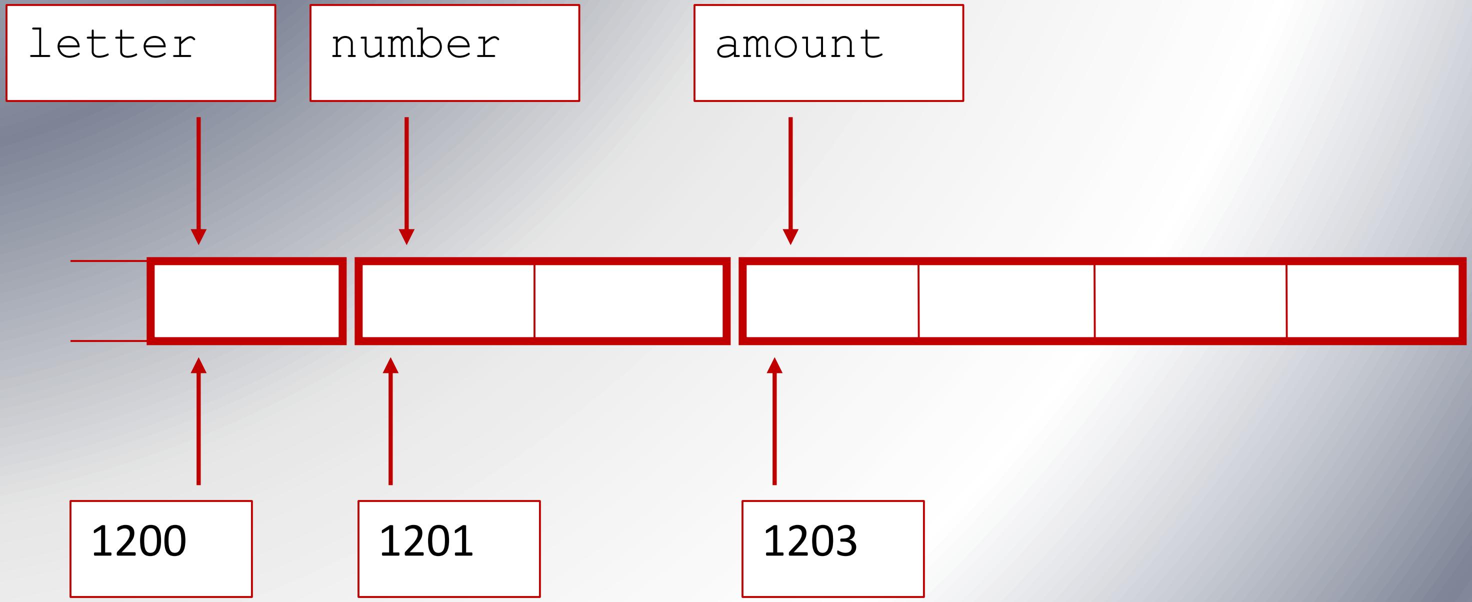
```cpp
// Fig. 4.4: fig04_04.cpp
// Initializing an array with a declaration.
#include <iostream>

using std::cout;
using std::endl;

#include <iomanip>

using std::setw;

int main()
{
  // use initializer list to initialize array n
  int n[ 10 ] = { 32, 27, 64, 18, 95, 14, 90, 70, 60, 37 };

  cout << "Element" << setw( 13 ) << "Value" << endl;

  // output contents of array n in tabular format
  for ( int i = 0; i < 10; i++ )
    cout << setw( 7 ) << i << setw( 13 ) << n[ i ] << endl;

  return 0;  // indicates successful termination

} // end main
```

Note the use of the initializer list.

```
Element        Value
      0           32
      1           27
      2           64
      3           18
      4           95
      5           14
      6           90
      7           70
      8           60
      9           37
```

Conditional ternary operator (?)

- The address operator (&) returns the memory address of a variable.



```
// This program uses the & operator to determine a variable’s
// address and the sizeof operator to determine its size.

#include <iostream.h>

void main(void)
{
   int x = 25;
   cout << "The address of x is " << &x << endl;
   cout << "The size of x is " << sizeof(x) << " bytes\n";
   cout << "The value in x is " << x << endl;
}
```

```
The address of x is 0x8f05
The size of x is 2 bytes
The value in x is 25
```

Conditional ternary operator (?)

A pointer is a variable that holds a memory address. That’s it.

- This is what the difference in between variable and pointer.
  - Pointer holds the address
  - Variable holds the value.

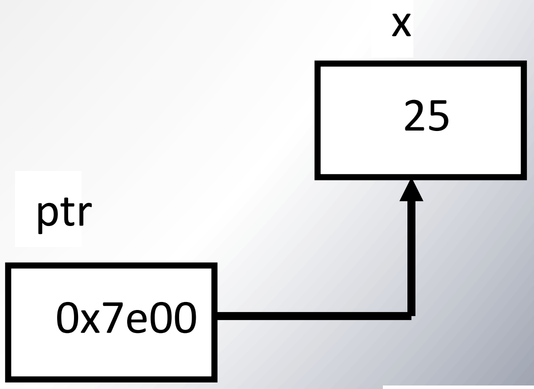
Computer memory is divided into sequentially numbered memory locations. Each variable is located at a unique location in memory, known as its address.

Pointers are useful for the following

- Working with memory locations that regular variables don’t give you access to
- Working with strings and arrays
- Creating new variables in memory while the program is running
- Creating arbitrarily-sized lists of values in memory

```
// This program stores the address of a variable in a
   pointer.
#include <iostream.h>

void main(void)
{
   int x = 25;
   int *ptr;

   ptr = &x;    // Store the address of x in ptr
   cout << "The value in x is " << x << endl;
   cout << "The address of x is " << ptr << endl;
}
```

```
The value in x is 25
The address of x is 0x7e00
```



Address of x:
0x7e00

```
// This program demonstrates the use of the indirection
// operator.
#include <iostream.h>

void main(void)
{
   int x = 25;
   int *ptr;

   ptr = &x;    // Store the address of x in ptr
   cout << "Here is the value in x, printed twice:\n";
   cout << x << "  " << *ptr << endl;
   *ptr = 100;
   cout << "Once again, here is the value in x:\n";
   cout << x << "  " << *ptr << endl;
}
```

```
Here is the value in x, printed twice:
25  25
Once again, here is the value in x:
100  100
```

Thanks!