Lab #2

1D Arrays

Structured Programming 2017/2018

Arrays

- Unlike regular variables, arrays can hold multiple values.
- Structures of related data items
- Consecutive group of memory locations
- Same name and type (**`int`**, **`char`**, etc.)
- Static entity (same size throughout program)

Arrays

To refer to an element:

- Specify array name and position number (index)
- Format: array name[position number]
- First element at position 0

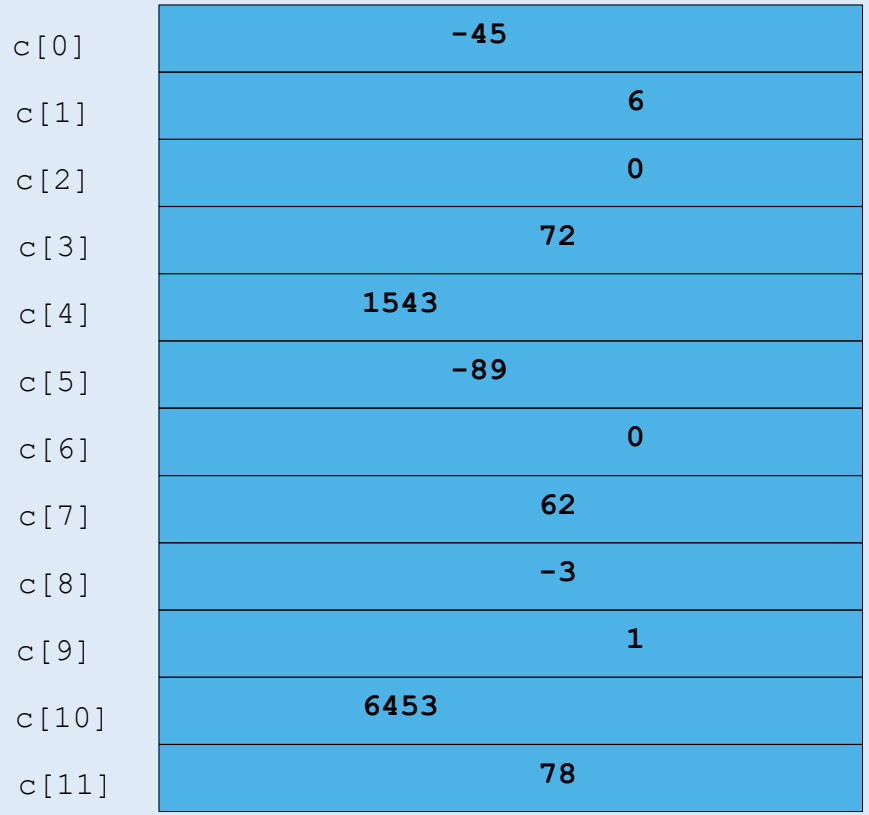
N-element array c

```
c[ 0 ], c[ 1 ] … c[ n - 1 ]
```

- Nth element as position N-1

Arrays

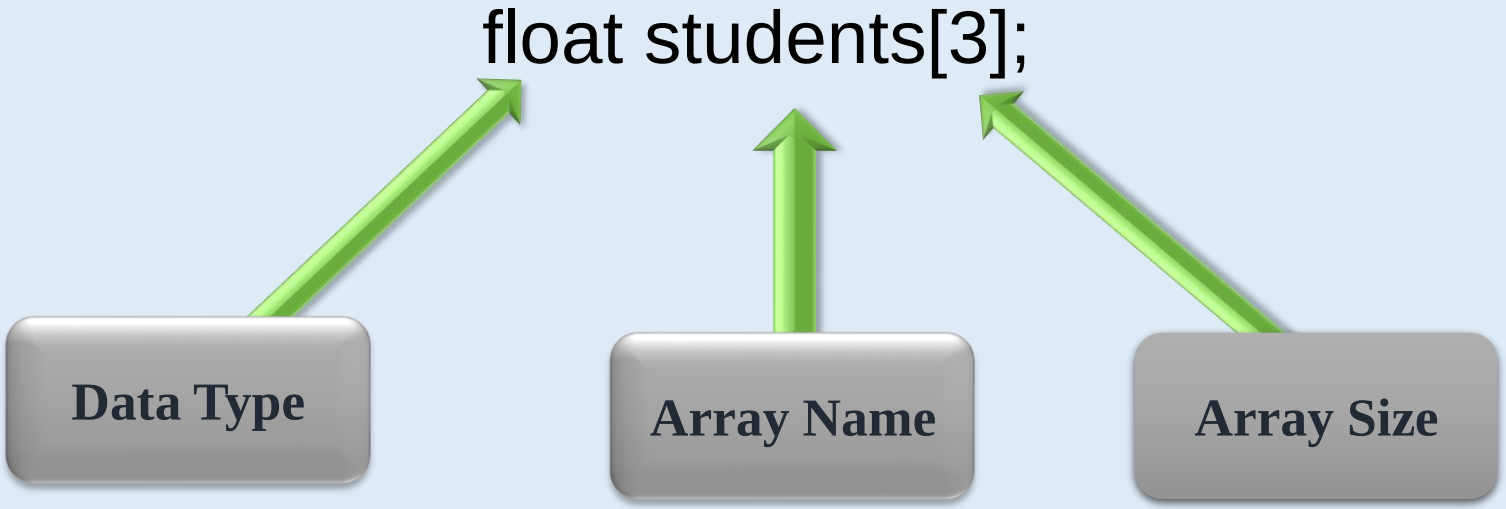
- Name of array (Note that all elements of this array have the same name, c)
- Position number of the element within array c



Array Declaration

When declaring arrays, specify:

- Name
- Type of array
  - Any data type
- Number of elements
- *type arrayName* **[** *arraySize* **];**




```
int c[ 10 ];  // array of 10 integers

float d[ 28 ]; // array of 28 floats
```

Array Declaration

- Declaring multiple arrays of same type
  - Use comma separated list, like regular variables

```
int b[ 100 ], x[ 27 ];
```

- Array size
  - Can be specified with constant variable (**const**)
    - **const int size = 20;**
  - Constants cannot be changed
  - Constants must be initialized when declared
  - Constants are also named read-only variables

Array Declaration

What will happen if the following code is executed?!

```
int main()
{
    int arraySize = 3;
    float students[arraySize];
}
```

The compiler will give an **error**, as *arraySize* is not constant at compile-time, so the compiler does not know how much space to allocate for ‘students’ array.

Array Declaration

What will happen if the following code is executed?!

```
int main()
{
const int arraySize = 3;

float students[arraySize];
}
```

The compiler will not give an error, as *arraySize* is constant at compile-time. A three-element float array will be created, and named ‘students’.

Array Declaration

What will happen if the following code is executed?!

```
#define arraySize 3

int main()

{

    float students[arraySize];

}
```

The compiler will not give an error, as *arraySize* is constant at compile-time. A three-element float array will be created, and named ‘students’.

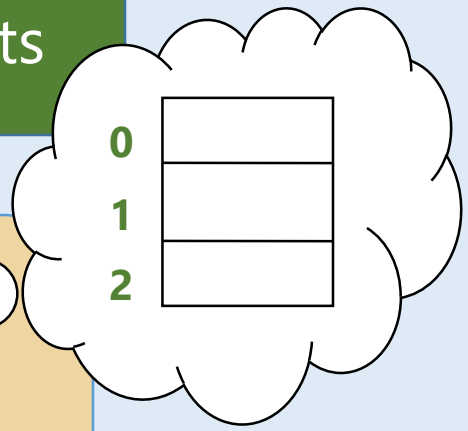
Single Element Vs Array(Code)

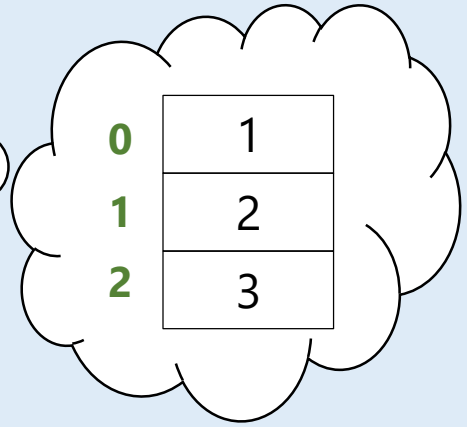
One Element

```
//declaration
int element;

//declaration &
  initialization
int element = 10;
```

Array of elements

```
//declaration
int elements[3];

//declaration &
  initialization
int elements[3] =
        {1,2,3};
```





Single Element Vs Array (Memory Allocation)

`int count` Enough memory for **1 `int`** | 12345

`float price` Enough memory for **1 `float`** | 56.981

`char letter` Enough memory for **1 `char`** | A

Array Declaration	Number of Elements	Size of Each Element	Size of the Array
char letters[25];	25	1 byte	25 bytes
short rings[100];	100	2 bytes	200 bytes
int miles[84];	84	4 bytes	336 bytes
float temp[12];	12	4 bytes	48 bytes
doubledDistance[1000];	1000	8 bytes	8000 bytes

Array Initialization

- For loop
  - Set each element of the array within a for loop

```
for (int i = 0; i < arraySize; i++)
    students[i]++;
```

- Initializer list
  - Specify each element when array declared

```
int n[ 5 ] = { 51, 22, 63, 34, 35 };
```

  - If not enough initializers, rightmost elements 0 (Partial Array Initialization)

```
int numbers[7] = {1, 2, 4, 8};
```

  - If too many , then a syntax error will occur

```
int numbers[3] = {10, 20, 30, 40 };
```

Array Initialization

- Set every element to value 0

```
int n[ 5 ] = { 0 };
```

- If array size omitted, initializers determine size

```
int n[] = { 91, 62, 13, 44, 25 };
```

  - 5 initializers, therefore 5 element array

Example 1: Input by the user

```
#include<iostream>

using namespace std;

void main()

{

    float hours[6];

    cout << "Enter the hours worked by
    six employees: ";

    cin >> hours[0];

    cin >> hours[1];

    cin >> hours[2];

    cin >> hours[3];

    cout << "The hours you entered are:";

    cout << " " << hours[0];

    cout << " " << hours[1];

    cout << " " << hours[2];

    cout << " " << hours[3];

}
```

Example 2: Input by the user (Another method)

```
#include<iostream>

using namespace std;

void main()

{

    float hours[6];

    cout << "Enter the hours worked by six
    employees: ";

    for (int count = 0; count < 6; count++)

        cin >> hours[count];

    cout << "The hours you entered are:";


    for (int count = 0; count < 6;
    count++)

        cout << " " << hours[count] <<
        endl;

}
```

Note

- Now that we know how to declare & Initialize arrays, we need to know how to get the value of an element stored at a particular index.
- This is done by subscripts, written in square brackets.
- Arrays are zero based, meaning that first element in the array is stored in index 0: *Students [0]*
- To refer to the element, for example, in the index 2 of the array students we would write:
- students[2] → Third element in the array

Note

- Variables may be used in subscripting as well.

```
int i = 0;

students[i] = students[i + 2];
```

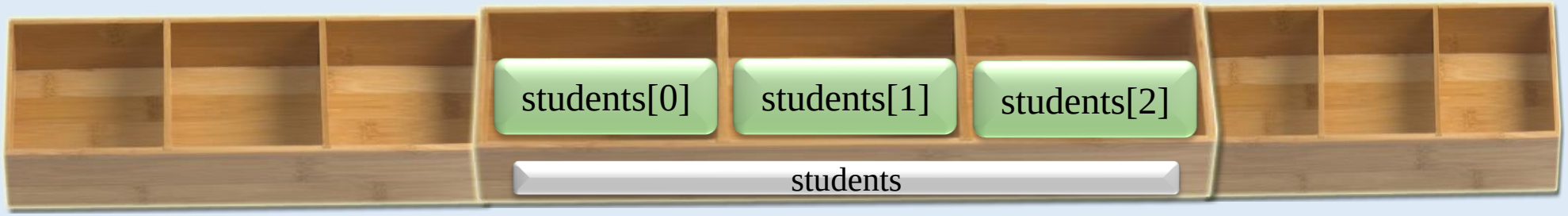
- The above code performs the same task that the following code does:

```
students[0] = students[2];
```

Note

int students[3];

- Referencing students[3] would result in an incorrect result since students[3] is outside the range of the array.
- The C++ compiler **will not** give you an error, but you will get unpredictable results if you try to access values outside the range of the array.



Example 3

- **For loops can be used to enter the values of the array elements, process them, and finally output their values.**

```
const int arraySize = 3;

int students[arraySize];

for (int i = 0; i < arraySize; i++)

    cin >> students[i];

for (int i = 0; i < arraySize; i++)

    students[i]++;

for (int i = 0; i < arraySize; i++)

    cout << students[i];
```

Store & Display

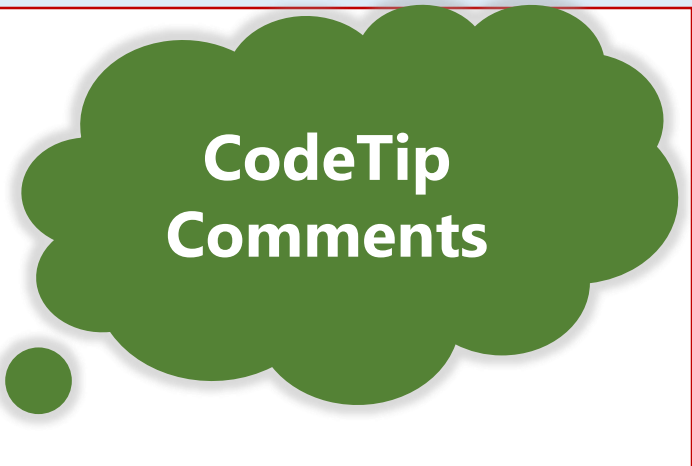
Write a C++ Program to store 5 numbers entered by user in an array and display first and last number only.

OUTPUT

```
Enter 5 numbers :
4
 -3
 5
 2
 0

 First number: 4
 Last number: 0
```

Store & Display: Solution

```cpp
#include<iostream>
using namespace std;
int main()
{
    int n[5];
    cout << "Enter 5 numbers: \n";
    /* Storing 5 number entered by user in an array using for loop. */
    for (int i = 0; i < 5; ++i)
    {
        cin >> n[i];
    }
    cout << "First number: " << n[0] << endl; // first element of an array is n[0]
    cout << "Last number: " << n[4] << endl;// last element of an array is n[4]
    return 0;
}
```



n[SIZE_OF_ARRAY - 1]

Square of 1D Array

Write a program that asks the user to enter a list of 5 numbers and outputs a list of the squares of these numbers.

OUTPUT

Please enter 5 numbers:
4
13
-2
0
8
The list of squares: 16 , 169, 4 , 0 , 64

Square of 1D Array: Solution

```
#include <iostream>
using namespace std;
void main()
{
  int arr[5]; //1. declare array
  cout<<"Please enter 5
  numbers"<<endl;

  //2. fill the input array from
  user
  for(int i = 0 ; i < 5; i++)
    cin>>arr[i];

  //3. Calculate the list of
  squares
  cout<<"The List of Squares: ";
  for(int i = 0 ; i < 5; i++)
  {
    arr[i]*=arr[i];
    cout<<arr[i]<<' ';
  }
  cout<<endl;
}
```

Exercises

Ready… Steady… Code!

Search for a Value

Write a program that asks the user to type 5 integers of an array and an integer value V. The program must search if the value V exists in the array and its location.

OUTPUT

```
Please enter 5 numbers:
0
-5
100
-1
8
Enter the value of V: 100

Value exists at location 2
```

Search for a Value: Solution

```
#include <iostream>
using namespace std;
void main()
{
   const int N = 5;
   int t[N], i, j, V;
   bool found = false;
   cout<<"Please enter 5 numbers"<<endl;

   for (i = 0; i<N; i++)
   {
      cin >> t[i];
   }
   cout << “Enter the value of V: ";
   cin >> V;
```

```
   for (i = 0; i < N; i++)
   {
      if (t[i] == V)
      {
         found = true;
         break;
      }
   }
   if(found)
      cout << “Value exists at location "
      << i << endl;
   else
      cout << “Value doesn’t exist." <<
      endl;

}
```

Student System

Calculate the Average of students grades

Write a Program that takes n number of students grades from user(where, n is specified by user but not more than 100), stores data in an array and calculates the average of those grades.

OUTPUT

```
Enter the numbers of grades: 6
Enter grade : 45.3
Enter grade : 67.5
Enter grade : -45.6
Enter grade : 20.34
Enter grade : 33 6
Enter grade : 45.6
Average grade= 27.69
```

Calculate the Average: Solution

```
#include <iostream>
using namespace std;
int main()
{
   int n, i;
   float num[100], sum = 0.0, average;
   cout << "Enter the numbers of data: ";
   cin >> n;
   while (n>100 || n <= 0)
   {
      cout << "Error! number should in
      range of (1 to 100)." << endl;
      cout << "Enter the number again: ";
      cin >> n;
   }
   for (i = 0; i<n; ++i)
   {
      cout << i + 1 << ". Enter number:
      ";
      cin >> num[i];
      sum += num[i];
   }
   average = sum / n;
   cout << "Average = " << average <<
   endl;
   return 0;
}
```

Minimum Value

Write a C++ program to accept from the user an array of 8 grades and display the minimum grade and its location.

OUTPUT

```
Please enter 8 numbers:
4
13
50
2
5
100
75
8

Min = 2 , at location 4
```

Minimum Value: Solution

```
#include <iostream>
using namespace std;
void main()
{
  const int N = 8;
  int i, arr[N], min, minindex;
  for (i=0; i<N; i++)
    cin>>arr[i];

  min = arr[0];
  minindex = 0;
  for (i=1; i<N; i++) {
    if (min > arr[i]) {
      min = arr[i];
      minindex = i;
    } // if
  } // for
  cout<<“Min = "<<min<<" , at
  location "<<minindex<<endl;

}
```

Thank you!