



Computer Programming (a)

E1123

Fall 2022-2023

Lecture 11

Arrays



INSTRUCTOR

DR / AYMAN SOLIMAN

➤ Contents

- 1) Introduction
- 2) Arrays
- 3) Accessing Array Components
- 4) Processing One-Dimensional Arrays



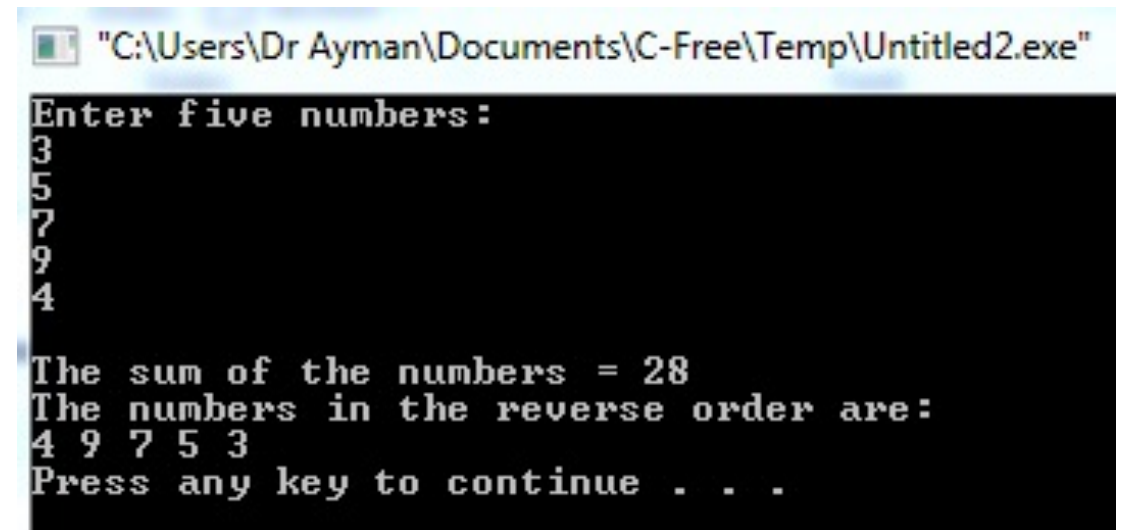
➤ Introduction

- **Before** defining an array, let us consider the **following problem**. We want to write a C++ program that reads five numbers, finds their sum, and prints the numbers in reverse order.
- Lastly, you learned how to read numbers, print them, and find the sum. The difference here is that we want to print the numbers in reverse order. To do this, we need to store all the numbers before we start printing them in reverse order. the following program accomplishes this task.

➤ Example Solving

//Program to read five numbers, find their sum, and print the numbers in reverse order.

```
#include <iostream>
using namespace std;
int main()
{
int num0, num1, num2, num3, num4;
int sum=0;
cout << "Enter five numbers: "<<endl;
cin >> num0 >> num1 >> num2 >> num3 >> num4;
cout << endl;
sum = num0 + num1 + num2 + num3 + num4;
cout << "The sum of the numbers = " << sum << endl;
cout << "The numbers in the reverse order are: "<<endl;
cout << num4 << " " << num3 << " " << num2 << " "
<< num1 << " " << num0 << endl;
return 0;
}
```



The screenshot shows a Windows command prompt window titled "C:\Users\Dr Ayman\Documents\C-Free\Temp\Untitled2.exe". The program prompts "Enter five numbers:" and the user enters the numbers 3, 5, 7, 9, and 4 on separate lines. The program then outputs "The sum of the numbers = 28" and "The numbers in the reverse order are: 4 9 7 5 3". Finally, it prompts "Press any key to continue . . .".

➤ Comments

- This program works fine. However, if you need to read 100 (or more) numbers and print them in reverse order, you will have to declare 100 variables and write many cin and cout statements.
- Note the following in the previous program:
 1. Five variables must be declared because the numbers are to be printed in reverse order
 2. All variables are of type (int—) that is, of the same data type.
 3. The way in which these variables are declared indicates that the variables to store these numbers all have the same name—except the last character, which is a number.

➤ Comments (cont.)

- **Statement 1** tells you that you must declare five variables.
- **Statement 3** tells you that it would be convenient if you could somehow put the last character, which is a number, into a **counter** variable and use one for **loop** to count **from 0 to 4** for reading and another for loop to count **from 4 to 0** for printing. Finally, because all variables are of the **same type**, you should be able to **specify how many variables must be declared** - and their data type - with a simpler statement than the one we used earlier.
- **The data structure that lets you do all these things in C++ is called an array.**

Arrays

```
graph TD; A[Arrays] --> B[one-dimensional]; A --> C[Two-dimensional]
```

one-dimensional

Two-dimensional

➤ Arrays

- An array is a **collection** of a **fixed number** of components all the **same data type**.
- A one-dimensional array is an array in which the components are arranged in a **list** form.
- Syntax for **declaring** a one-dimensional array:

```
dataType arrayName[intExp];
```

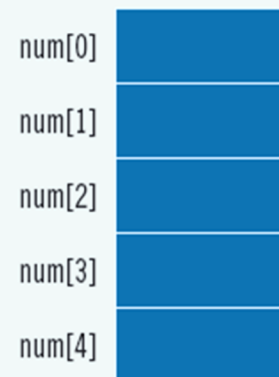
- In which **intExp** is any constant expression that evaluates to a **positive integer**.
- Also, **intExp** specifies the number of components in the array.

➤ Example 1

➤ The statement:

```
int num[5];
```

declares an array num of five components. Each component is of type int. The components are num[0], num[1], num[2], num[3], and num[4]. Next Figure illustrates the array num.



➤ Accessing Array Components

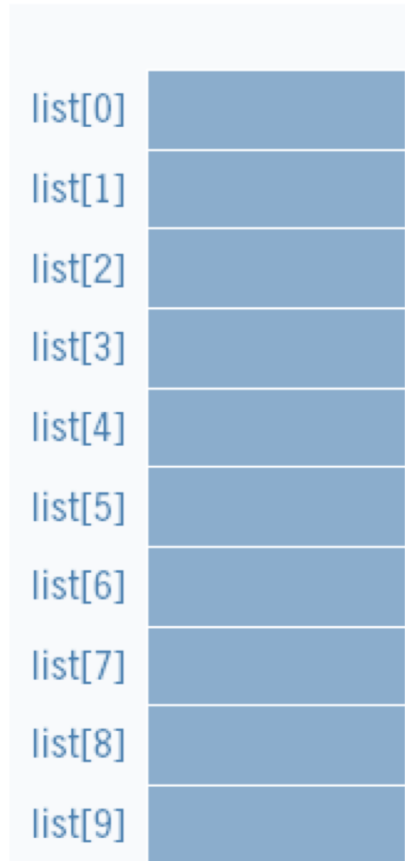
➤ General syntax:

```
arrayName[indexExp]
```

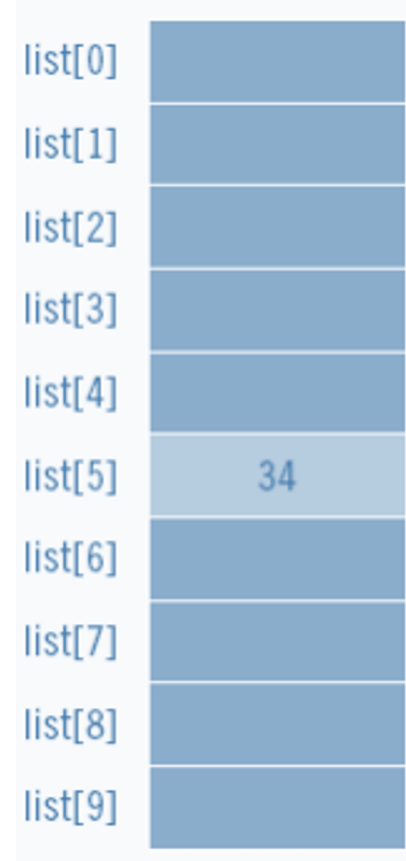
- where **indexExp** is called an **index**, is any expression whose value is a **nonnegative** integer
- Index value specifies the **position** of the component in the array
- [] is the array **subscripting operator**
- The array index always **starts at 0**

➤ Accessing Array Components (cont.)

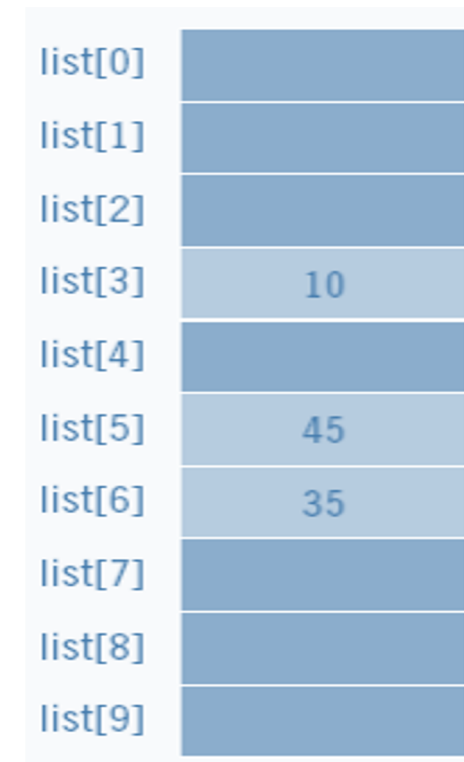
`Int list[10];`



`List[5]=34;`



`List[3]=10;`
`List[6]=35;`
`List[5]=list[3]+list[6];`



➤ Processing One-Dimensional Arrays

- Some **basic operations** performed on a one-dimensional array are:
 - Initializing
 - Inputting data
 - Outputting data stored in an array
 - Finding the largest and/or smallest element
- Each operation requires ability to step through the elements of the array.
- Easily accomplished by a loop.

➤ Processing One-Dimensional Arrays (cont.)

- Consider the declaration

```
int list[100];           //array of size 100  
int i;
```

- Using for loops to access array elements:

```
for (i = 0; i < 100; i++) //Line 1  
process list[i]           //Line 2
```

- Example:

```
for (i = 0; i < 100; i++) //Line 1  
cin >> list[i];           //Line 2
```

➤ Example 2

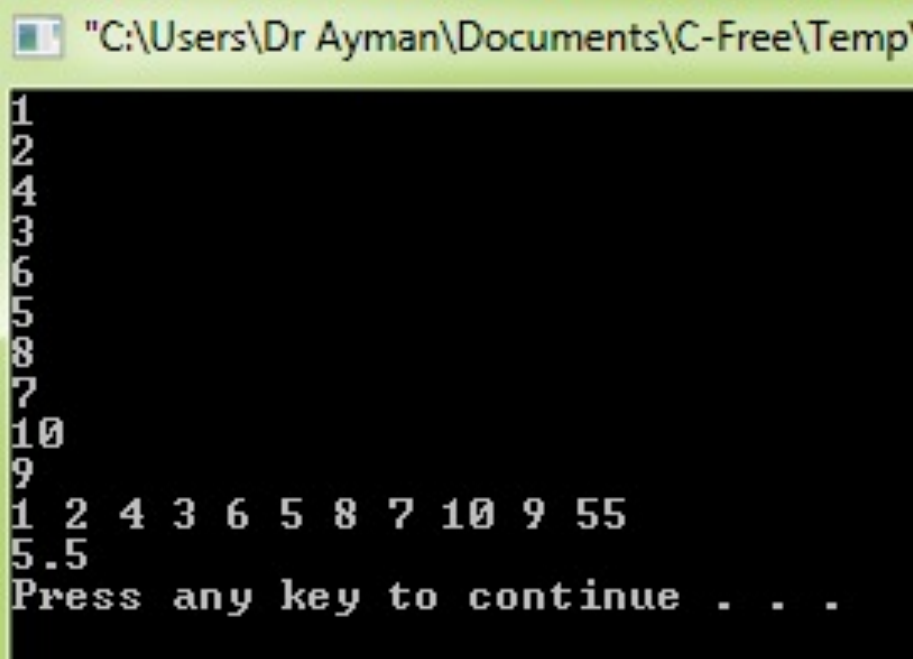
- Finding the **sum** and **average** in the array (Printing the array)

- The steps of solution
 - a. Initializing an array
 - b. Reading data into an array
 - c. Printing an array
 - d. Finding the sum and average of the array

➤ Example 2 solving

```
#include <iostream>
using namespace std;
int main()
{
double list[10];
int i;
double sum=0, average;
for (i = 0; i < 10; i++)
{
list[i] = 0;
cin >> list[i];
}
for (i = 0; i < 10; i++)
cout << list[i] << " ";
```

```
for (i= 0;i < 10; i++)
{
sum = sum + list[i];
average = sum / 10;
}
cout<<sum<<endl;
cout<<average<<endl;
}
```



The screenshot shows a Windows command prompt window with the title bar "C:\Users\Dr Ayman\Documents\C-Free\Temp". The program has been executed, and the output is displayed on a black background with white text. The output consists of the numbers 1 through 10 entered on separate lines, followed by a space-separated list of the same numbers: "1 2 4 3 6 5 8 7 10 9 55". Below this, the sum "55" is printed, followed by the average "5.5". The prompt "Press any key to continue . . ." is visible at the bottom.

➤ Example 3

- Finding the **smallest** and **Largest** element in the array (Printing the array in reverse order)

- The steps of solution
 - a. Initializing an array
 - b. Reading data into an array
 - c. Printing an array in reverse order
 - d. Finding the smallest and largest of the array

➤ Example 3 solving

```
#include <iostream>
using namespace std;
int main()
{
double list[10];
int i,min,max;
double smallestlist,largestlist;
for (i = 0; i < 10; i++)
{
list[i] = 0;
cin >> list[i];
}

for (i = 9; i >= 0; i--)
```

```
max = 0;
for (i = 1; i < 10; i++)
{
if(list[max]<= list[i])
{
max = i;
}
largestlist = list[max];
}
cout<<largestlist<<endl;
```

```
min= 0;
for (i = 1; i < 10; i++)
{
if(list[min]>= list[i])
{
min = i;
}
smallestlist = list[min];
}
cout<<smallestlist<<endl;
}
```

➤ Example 4

➤ Write C++ program to calculate the interest of 10 customers as following:

income	interest
0>income<1000	0%
1000>income<10000	3%
10000>income<50000	5%
50000>income<100000	10%
100000>income<1000000	15%

➤ Example 4

```
#include <iostream>
using namespace std;
int main()
{
double list[10];
int i;
for (i = 0; i < 10; i++)
{
list[i] = 0;
cout<<"customer"<<i<<"=";
cin >> list[i];
}
for (i = 0; i <10; i++)
{
if (list[i]>=0&&list[i]<1000)
{
cout<<"The interest of customer"<<i<<"=0%"<<endl;
}
}
```

```
if (list[i]>=1000&&list[i]<10000)
{
cout<<"The interest of customer"<<i<<"=3%"<<endl;
}
if (list[i]>=10000&&list[i]<50000)
{
cout<<"The interest of customer"<<i<<"=5%"<<endl;
}
if (list[i]>=50000&&list[i]<100000)
{
cout<<"The interest of customer"<<i<<"=10%"<<endl;
}
if (list[i]>=100000&&list[i]<=1000000)
{
cout<<"The interest of customer"<<i<<"=15%"<<endl;
}
}
```

*Thank
you*

