Notes

• Please remember to write down your name and section number on your homework and project
Arrays
Objectives

- Arrays
  - What is an array?
  - How do you declare and initialize an array?
  - How can you use an array?
- Array Examples
- Array Practice
Example

**Question**: write a C program that reads the grades of 5 students and prints the number of students whose grade is less than average?
Example

Question: write a C program that reads the grades of 5 students and prints the number of students whose grade is less than average?

Two steps:

1- Read the grades and compute the average

2- Compare the individual grades to average one-by-one and count the number of them less than average
Example

```c
#include <stdio.h>

int main()
{
    float grade, average, sum, belowTheAverage;

    /* Step 1: Find the average of grades */

    /* Step 2: Compare the grades to average */
}
```
```c
#include <stdio.h>

int main()
{
    float grade, average, sum, belowTheAverage;

    /* Step 1: Find the average of grades */
    sum = 0.0;
    printf("Please enter the grades one by one: \n");
    for (int i = 0; i < 5; ++i)
    {
        scanf("%f", &grade);
        sum += grade;
    }
    average = sum/5;
    printf("The average is: %f\n", average);

    /* Step 2: Compare the grades to average */
    }
```
```c
#include <stdio.h>

int main()
{
    float grade, average, sum, belowTheAverage;

    /* Step 1: Find the averages of grades */
    sum = 0.0;
    printf("Please enter the grades one by one: \
");
    for (int i = 0; i < 5; ++i)
    {
        scanf("%f", &grade);
        sum += grade;
    }
    average = sum/5;
    printf("The average is: %f\n", average);

    /* Step 2: Compare the grades to average */

    return 0;
}
```
Example

```c
#include <stdio.h>

int main()
{
    float grade, average, sum, belowTheAverage;

    /* Step 1: Find the average of grades */
    sum = 0.0;
    printf("Please enter the grades one by one: \n");
    for (int i = 0; i < 5; ++i)
    {
        scanf("%f", &grade);
        sum += grade;
    }
    average = sum/5;
    printf("The average is: %f\n", average);

    /* Step 2: Compare the grades to average */

    All the grades are gone!
```
Example

```c
#include <stdio.h>

int main()
{
  float average, sum, belowTheAverage;
  float studMark1, studMark2, studMark3, studMark4, studMark5;

  /* Step 1: Find the average of grades */

  /* Step 2: Compare the grades to average */
```
Example

```c
#include <stdio.h>

int main()
{
    float average, sum, belowTheAverage;
    float studMark1, studMark2, studMark3, studMark4, studMark5;

    /* Step 1: Find the average of grades */
    sum = 0.0;
    printf("Please enter the grades one by one: \n");
    scanf("%f", &studMark1);
    sum += studMark1;
    scanf("%f", &studMark2);
    sum += studMark2;
    scanf("%f", &studMark3);
    sum += studMark3;
    scanf("%f", &studMark4);
    sum += studMark4;
    scanf("%f", &studMark5);
    sum += studMark5;
    average = sum/5;
    printf("The average is: \f\n", average);

    /* Step 2: Compare the grades to average */
}
```

Can **NOT** use loop structures!
```c
#include <stdio.h>

int main()
{
    float average, sum, belowTheAverage;
    float studMark1, studMark2, studMark3, studMark4, studMark5;

    /* Step 1: Find the average of grades */
    sum = 0.0;
    printf("Please enter the grades one by one: \n");
    scanf("%f", &studMark1);
    sum += studMark1;
    scanf("%f", &studMark2);
    sum += studMark2;
    scanf("%f", &studMark3);
    sum += studMark3;
    scanf("%f", &studMark4);
    sum += studMark4;
    scanf("%f", &studMark5);
    sum += studMark5;
    average = sum/n;
    printf("The average is: %f\n", average);

    /* Step 2: Compare the grades to average */
    belowTheAverage = 0;
    if(studMark1 < average) belowTheAverage++;
    if(studMark2 < average) belowTheAverage++;
    if(studMark3 < average) belowTheAverage++;
    if(studMark4 < average) belowTheAverage++;
    if(studMark5 < average) belowTheAverage++;
    printf("The number of grades less than average is: %f\n", belowTheAverage);
}
```
Example

Question: write a C program that reads the grades of 1000 students and prints the number of students whose grade is less than average?
Example

**Question:** write a C program that reads the grades of **1000** students and prints the number of students whose grade is less than average?

- Will you define 1000 variables?

```c
int studMark1, studMark2, studMark3,
studMark4, ..., ..., studMark998, stuMark999,
studMark1000;
```

- Is there any better idea than declaring many single variables?
Data Types

• Scalar Types: hold a single a value
  – float
  – int
  – char, ...

• Aggregate Data Types: referenced as a single entity but hold several values
  – array
  – struct
Single versus Aggregate Data Types

不像普通变量，数组可以存储多个值。

- `int count` 足够存储 1 个 int 类型的内存
  - 12345

- `float price` 足够存储 1 个 float 类型的内存
  - 56.981

- `char letter` 足够存储 1 个 char 类型的内存
  - A

不像普通变量，数组可以存储多个值。
An array is a group of related data items that all have the same name and the same data type.

Arrays can be of any data type we choose.

An array’s data items are stored contiguously in memory.

Each of the data items is known as an element of the array.

Each element can be accessed individually.
- **One Dimensional Array**

```
5  2  6  9  3
```

- **Two-Dimensional Arrays**

```
<table>
<thead>
<tr>
<th>column 0</th>
<th>column 1</th>
<th>column 2</th>
<th>column 3</th>
<th>column 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>row 1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>row 2</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
```

- (general) **Multidimensional Arrays ( > 2)**
Declaring Arrays
Array Declaration

When declaring arrays, specify
- Name
- Type of array
- Number of elements

\[ \text{array\_Type arrayName[ numberOfElements ];} \]

The type of individual elements: int, float, char, etc

The number of elements you want to store in the array
Array Declaration

```
arrayType arrayName[ numberOfElements ];
```

Examples:

```
int list[10];
```

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
Example

```c
arrayType arrayName[ numberOfElements ];
int list [ 10 ];
```

- The name of this array is “list”.
- The type of individual elements is `int`
- This declaration sets aside a chunk of memory that is big enough to hold 10 integers.
- It **does NOT initialize** those memory locations to 0 or any other value. They contain garbage.
More Examples

arrayType arrayName[numberOfElements];

int StudentsName[1000];

float scores[20];

char name[10];
arrayType arrayName[ numberOfElements ];

int list [ -5 ];

Not To Do

int list [ 2.5 ];
arrayType arrayName[ numberOfElements ];

int list[ ];

![Error message from compiler](image-url)
Initializing Arrays
Initializing Array Elements

An array may be initialized (optional) at the time of declaration.

```plaintext
arrayType arrayName[numberOfElements] = {a_list_of_values};
```

- The value of elements you want to store in the array.
- Separate elements by comma.
An array may be **initialized at the time of declaration.**

```c
arrayType arrayName[numberOfElements] = {a_list_of_values};
```

```c
int numbers [5] = {5, 2, 6, 9, 3};
```

**numbers** $\Rightarrow$ 5 2 6 9 3

- The value of elements you want to store in the array.
- Separate elements by comma.
Initializing Array Elements

An array may be **initialized at the time of declaration.**

```
arrayType arrayName[numberOfElements] = {a_list_of_values};
```

- If initializer is **shorter than array size**, the remainder of the array (rightmost elements) is initialized to zero:
  ```
  int numbers[10] = {5, 2, 6, 9, 3};
  ```

```
  5 2 6 9 3 0 0 0 0 0
```

- If initializer is longer than array size: **error**
Initializing and Size of the Array

arrayType arrayName[numberOfElements] = {a_list_of_values};

• It is NOT necessary to define the size of arrays during initialization.
Initializing and Size of the Array

arrayType arrayName[numberOfElements] = {a_list_of_values},

• It is **NOT** necessary to define the size of arrays during initialization.

• In this case, the compiler determines the size of array by calculating the number of elements of an array (Implicit Array Sizing)
### Initializing and Size of the Array

- It is **NOT** necessary to define the size of arrays during initialization.

- In this case, the compiler determines the size of array by calculating the number of elements of an array.

```c
int numbers[] = {1, 2, 3, 4, 5} = int numbers[5] = {1, 2, 3, 4, 5};
```

```
5 2 6 9 3
```

```c
int scores[100] = {0} // assigns zero to all 100 elements
```

```
0 0 0 0 0 0
```
Arrays with Char Data Type

- Initialization of an array of type char take the following form,

  ```c
  char    array_name[size] = "string_lateral_constant";
  ```

- For example, the array `name` can be initialized as follows:

  ```c
  char name[6] = "abcde" == char name[6] = {a,b,c,d,e}
  ```

- For unsized array (variable sized), we can declare as follow,

  ```c
  char chName[ ] = "Mr. Dracula";
  ```

C compiler automatically creates an array which is big enough to hold all the initializer.
Accessing Array Elements
Accessing Array Elements

An element of an array is accessed using the array name and an index or subscript.

\[
\text{arrayName[index_of_element]};
\]

\[
\text{int a[10] = \{5, 2, 6, 9, 3\};}
\]
Accessing Array Elements

• “IMPORTANT”

• Array indexing in C starts at 0
  • If size \( n \), last index is \( n-1 \)

• int \( a[8] \)

\[
\begin{array}{c|c|c|c|c|c}
\hline
a & 11 & 20 & 0 & -2 \\
\hline
\hline
\end{array}
\]
Individual Elements

Use each individual element as a single variable:

```
//Set values
int i = 1
a[0] = 11;
a[1] = 20;
a[3+1] = 0;

//Print cell content
printf("%d\n", a[0]);

//Read into a cell
scanf("%d\n", &a[6]);
```
Assigning values to an array

For loops are often used to assign values to an array

Example:
```c
int list[5], i;

for(i=0; i<5; i++){
    list[i] = i;
}
```

OR

```c
for(i=0; i<=4; i++){
    list[i] = i;
}
```
Example

• Read 10 integers from the user and print them backwards: