CS2211b

Software Tools and Systems Programming

Week 9b

Arrays Part 1

Arrays

Arrays

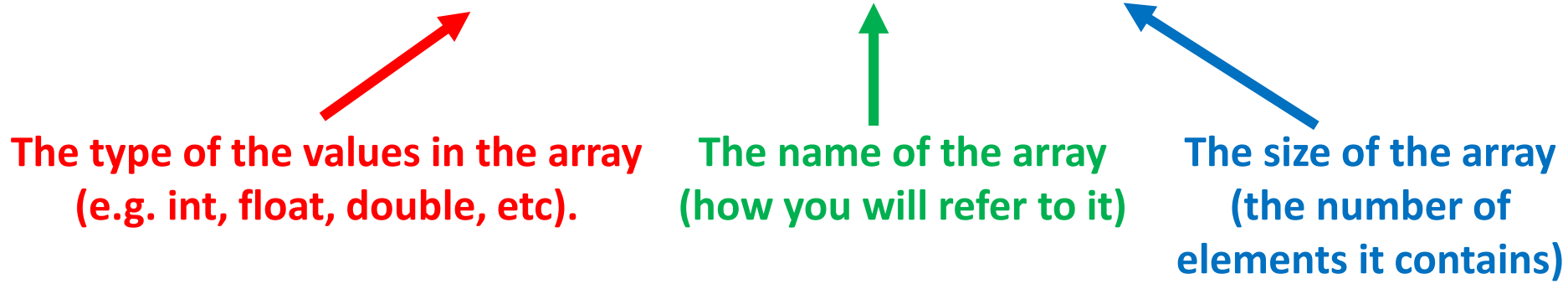
- So far all of the values and variables we have seen in C are **scalar**. That is, they are only capable of holding a single data value.
- C also supports **aggregated** variables, which can store a collection of values.
- There are two types of aggregated variables in C:
  - **Arrays**
  - **Structures**
- Today we will be talk about arrays.
- Arrays store a collection of one or more values of the **same type**.
- Each value (element) in the array is accessible by a unique index, their position in the array starting at 0.

Arrays

Syntax

Arrays are declared similarly to variables with the following syntax:

data_type array_name[size];



Simple Examples:

```
int a[10];
```
Array of 10 integers

```
char b[5];
```
Array of 5 characters

```
float c[42];
```
Array of 42 floating point numbers

Arrays

C89 vs. C99

In C89 the size of an array must be a constant value and know at compile time. You can not use a regular variable as the size of an array (e.g. int a[n]; is not allowed if n is not a constant).

In C99 you are allowed to have variable length arrays whose size are defined by a variable at run time. For example:

```
int n;
scanf("%d", &n);
int a[n];
```

Th

```
int a[10];
```
Array of 10 integers

```
char b[5];
```
Array of 5 characters

```
float c[42];
```
Array of 42 floating point numbers

Arrays

Syntax

We can access a specific element of an array using **subscript notation**:

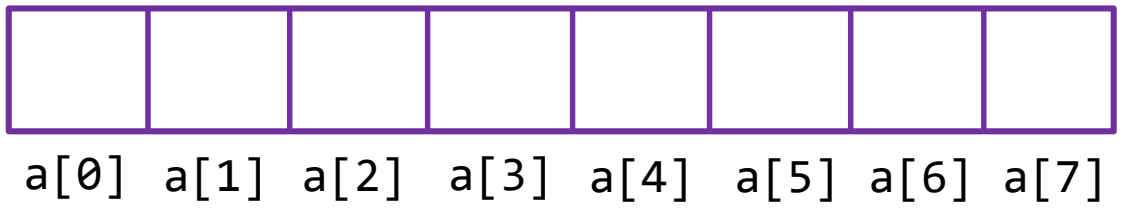
array_name[index]

The name of the array

The elements position in the array (starting at 0)

Example:

```
int a[8];
```



Arrays

Syntax

We can access a specific element of an array using **subscript notation**:

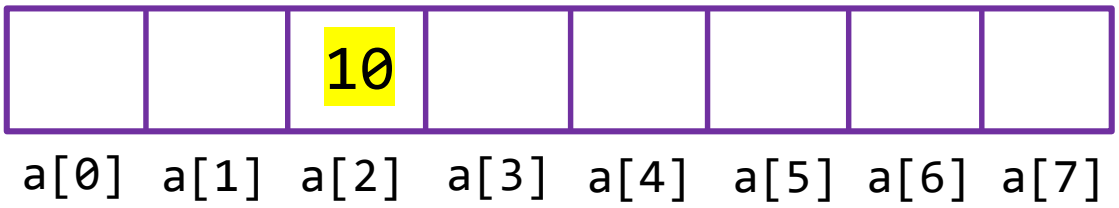
array_name[index]

The name of the array

The elements position in the array (starting at 0)

Example:

```
int a[8];
a[2] = 10;
```



Arrays

Syntax

We can access a specific element of an array using **subscript notation**:

array_name[index]

The name of the array

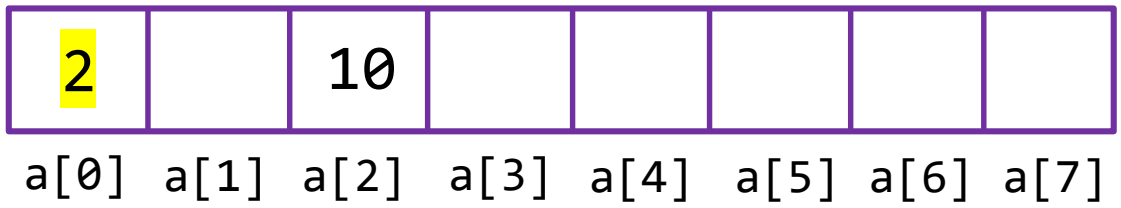
The elements position in the array (starting at 0)

Example:

```
int a[8];

a[2] = 10;

a[0] = 2;
```



Arrays

Syntax

We can access a specific element of an array using **subscript notation**:

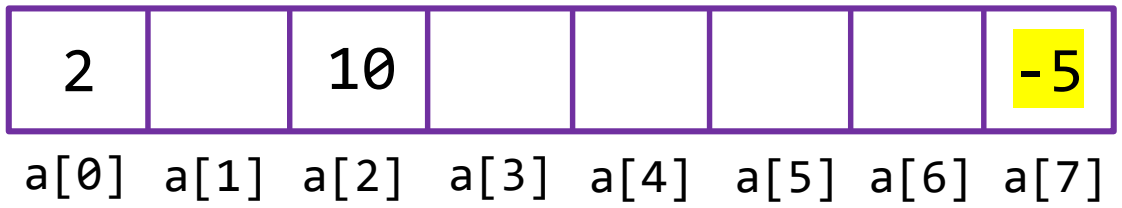
array_name[index]

The name of the array

The elements position in the array (starting at 0)

Example:

```
int a[8];
a[2] = 10;
a[0] = 2;
a[7] = -5;
```



Arrays

Syntax

We can access a specific element

```
printf("%d", a[0]);
```

would output 2

The name of the array

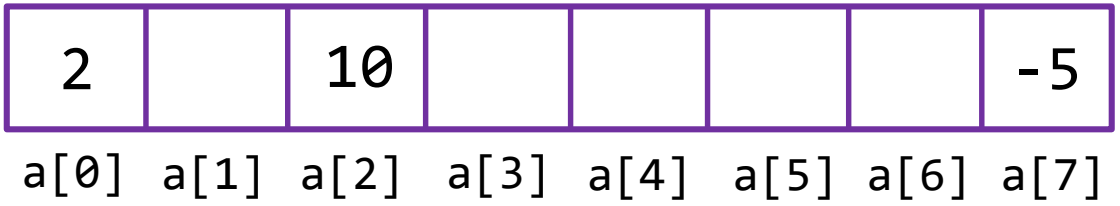
```
printf("%d", a[3]);
```

would be undefined behaviour as a[3] is not yet initialized.

array (starting at 0)

Example:

```
int a[8];
a[2] = 10;
a[0] = 2;
a[7] = -5;
```



Arrays

Syntax

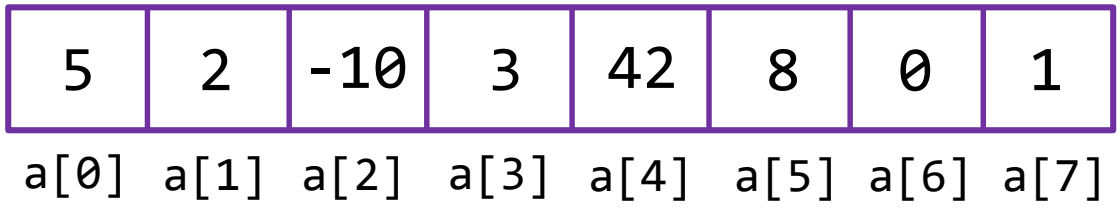
As a shortcut we can use the following notation to initialize an array on the same line it is declared:

data_type array_name[size] = {value1, value2, …, valueN };

Values in order they will appear in the array.

Example:

```
int a[8] = {5, 2, -10, 3, 42, 8, 0, 1};
```



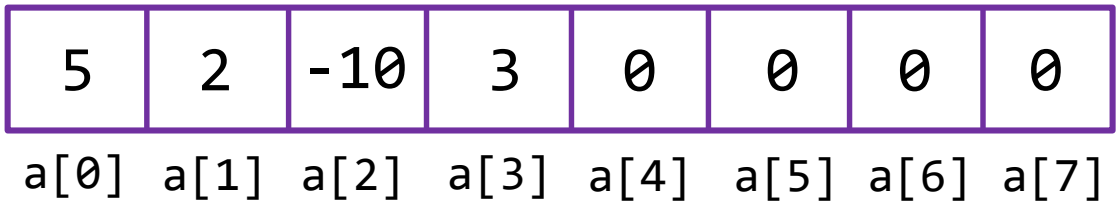
Arrays

Syntax

If we give less values then there are elements in the array, the remaining elements will be set to 0.

Example:

```
int a[8] = {5, 2, -10, 3};
```



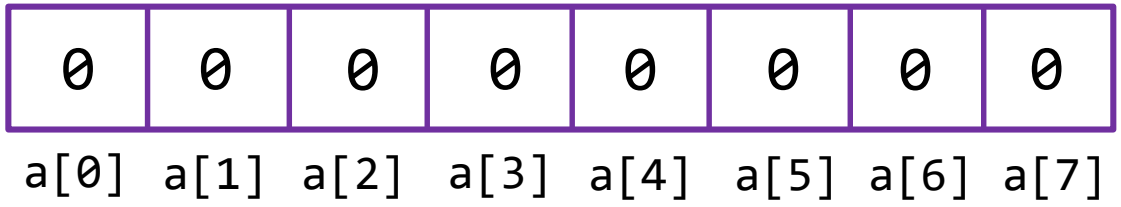
Arrays

Syntax

Can be useful when setting an array to all zero values.

Example:

```
int a[8] = {0};
```



Note that this will not work for any other number.

```
int a[8] = {1};
```

would only set the first element (a[0]) to 1 and the remaining elements would be 0.

Arrays

Syntax

We can not use this notation after an array has been declared.

Bad Example:

```
int a[8];
a = {1, 2, 3, 4, 5, 6, 7, 8};
```

This would give a syntax error when compiled.

Arrays

Syntax

We can not use this notation after an array has been declared

C99

C99 supports some additional ways to initialize your arrays. For example:

```
int a[8] = {[2] = 10, [5] = -3};
```

You can read about them in your C textbook (chapter 8.1, pages 165 to 166).

when compiled.

Arrays Example

/cs2211/week9/ex11.c

Input 5 integers from the user into an array then find the mean and standard deviation.

```c
#include <stdio.h>
#include <math.h>

#define SIZE 5

int main() {
    int nums[SIZE];
    int i;

    for(i = 0; i < SIZE; i++) {
        printf("Input number: ");
        scanf("%d", &nums[i]);
    }

    int sum = 0;
    int sqsum = 0;
    for(i = 0; i < SIZE; i++) {
        sum += nums[i];
        sqsum += nums[i] * nums[i];
    }

    float mean = (float) sum / SIZE;
    float stdev = sqrt(((float) sqsum / SIZE) - (mean * mean));

    printf("mean = %.2f\nstdev = %.2f\n", mean, stdev);
    return 0;
}
```

Arrays Example

/cs2211/week9/ex11.c

Input 5 integers from the user into an array then find the mean and standard deviation.

```c
#include <stdio.h>
#include <math.h>

#define SIZE 5

int main() {
    int nums[SIZE];
    int i;

    for(i = 0; i < SIZE; i++) {
        printf("Input number: ");
        scanf("%d", &nums[i]);
    }

    int sum = 0;
    int sqsum = 0;
    for(i = 0; i < SIZE; i++) {
        sum += nums[i];
        sqsum += nums[i] * nums[i];
    }

    float mean = (float) sum / SIZE;
    float stdev = sqrt(((float) sqsum / SIZE) - (mean * mean));

    printf("mean = %.2f\nstdev = %.2f\n", mean, stdev);
    return 0;
}
```

It is often good practice to use the #define directive to make a constant for your array size. This makes it easy to change later without editing much code.

Arrays Example

/cs2211/week9/ex11.c

Input 5 integers from the user into an array then find the mean and standard deviation.

```c
#include <stdio.h>
#include <math.h>

#define SIZE 5

int main() {
    int nums[SIZE];
    int i;

    for(i = 0; i < SIZE; i++) {
        printf("Input number: ");
        scanf("%d", &nums[i]);
    }

    int sum = 0;
    int sqsum = 0;
    for(i = 0; i < SIZE; i++) {
        sum += nums[i];
        sqsum += nums[i] * nums[i];
    }

    float mean = (float) sum / SIZE;
    float stdev = sqrt(((float) sqsum / SIZE) - (mean * mean));

    printf("mean = %.2f\nstdev = %.2f\n", mean, stdev);
    return 0;
}
```

Create an integer array of size 5 (as SIZE is defined as 5).

Arrays Example

/cs2211/week9/ex11.c

Input 5 integers from the user into an array then find the mean and standard deviation.

```c
#include <stdio.h>
#include <math.h>

#define SIZE 5

int main() {
    int nums[SIZE];
    int i;

    for(i = 0; i < SIZE; i++)
        printf("Input number:
        scanf("%d", &nums[i]);
    }

    int sum = 0;
    int sqsum = 0;
    for(i = 0; i < SIZE; i++) {
        sum += nums[i];
        sqsum += nums[i] * nums[i];
    }

    float mean = (float) sum / SIZE;
    float stdev = sqrt(((float) sqsum / SIZE) - (mean * mean));

    printf("mean = %.2f\nstdev = %.2f\n", mean, stdev);
    return 0;
}
```

This for loop runs 5 times (`i` = 0 to `i` = 4) and each iteration asks the user for a number and stores it in the next element of the `nums` array.

For loops are ideal for setting or running through the values of an array.

Arrays Example

/cs2211/week9/ex11.c

Input 5 integers from the user into an array then find the mean and standard deviation.

```c
#include <stdio.h>
#include <math.h>

#define SIZE 5

int main() {
    int nums[SIZE];
    int i;

    for(i = 0; i <
        printf("Inp
        scanf("%d",
    }

    int sum = 0;
    int sqsum = 0;
    for(i = 0; i < SIZE; i++) {
        sum += nums[i];
        sqsum += nums[i] * nums[i];
    }

    float mean = (float) sum / SIZE;
    float stdev = sqrt(((float) sqsum / SIZE) - (mean * mean));

    printf("mean = %.2f\nstdev = %.2f\n", mean, stdev);
    return 0;
}
```

We loop through the values of the nums array and create a sum of the values (for finding the mean) as well as the square sum of the values (for finding the standard deviation).

Arrays Example

/cs2211/week9/ex11.c

Input 5 integers from the user into an array then find the mean and standard deviation.

```c
#include <stdio.h>
#include <math.h>

#define SIZE 5

int main() {
    int nums[SIZE];
    int i;

    for(i = 0; i < SIZE; i++) {
        printf("Input number: ");
        scanf("%d", &nums[i]);
    }

    int sum = 0;
    int sqsum = 0;
    for(i = 0; i < SIZE; i++) {
        sum += nums[i];
        sqsum += nums[i] * nums[i];
    }

    float mean = (float) sum / SIZE;
    float stdev = sqrt(((float) sqsum / SIZE) - (mean * mean));

    printf("mean = %.2f\nstdev = %.2f\n", mean, stdev);
    return 0;
}
```

Calculate the mean and standard deviation based on the sums we calculated in the loop.

Arrays

Find the Size of an Array

- C does not provide an easy method of finding the size of an array. However, we can use a bit of math and the `sizeof` operator to calculate the size.
- Recall that the `sizeof` operator returns the size of a variable or data type in bytes.
- **Example:**

```
int a[5];
printf("%d %d\n", sizeof(a), sizeof(a[0]));
```

- **Output:**

```
20 4
```

Arrays

Find the Size of an Array

- C does not provide an easy method of finding the size of an array. However, we can use a bit of math and the `sizeof` operator to calculate the size.
- Recall that the `sizeof` operator returns the size of a variable or data type in bytes.
- **Example:**

```
int a[5];
printf("%d %d\n", sizeof(a), sizeof(a[0]));
```

- **Output:**

```
20 4
```

`sizeof(a[0])` gives the size of a single element of the array a. In this case this is an integer that takes up 4 bytes.

Arrays

Find the Size of an Array

- C does not provide an easy method of finding the size of an array. However, we can use a bit of math and the `sizeof` operator to calculate the size.
- Recall that the `sizeof` operator returns the size of a variable or data type in bytes.
- **Example:**

```
int a[5];
printf("%d %d\n", sizeof(a), sizeof(a[0]));
```

- **Output:**

```
20 4
```

`sizeof(a)` gives the size of the whole array in bytes. In this case, an integer is 4 bytes so an array of 5 integers is 20 bytes (4 x 5 = 20).

Arrays

Find the Size of an Array

- To get the number of elements in the array we simply have to divide the size of the array in bytes by the size of an individual element in bytes.
- **Example:**

```
int a[5];
printf("%d\n", sizeof(a) / sizeof(a[0]));
```

- **Output:**

```
5
```