Practical Programming

The C Language : Arrays and Strings

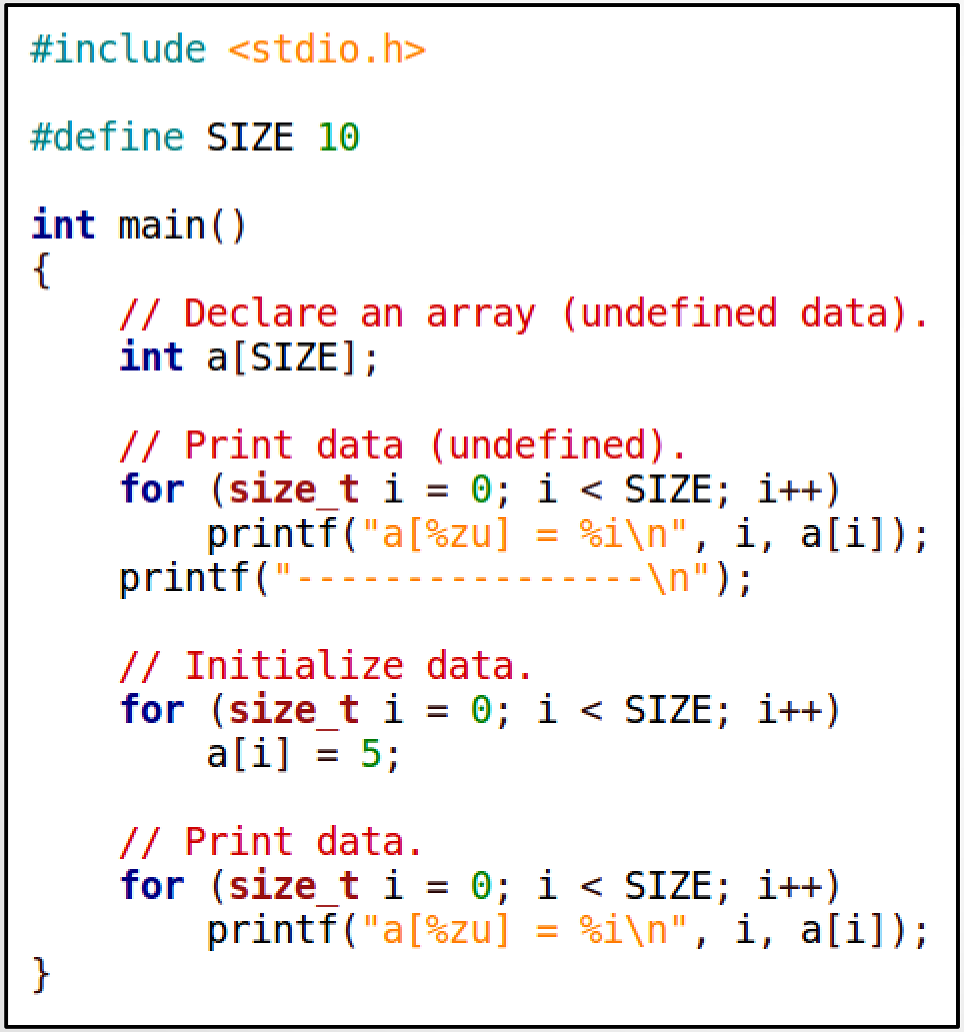
David Bouchet

david.bouchet.epita@gmail.com

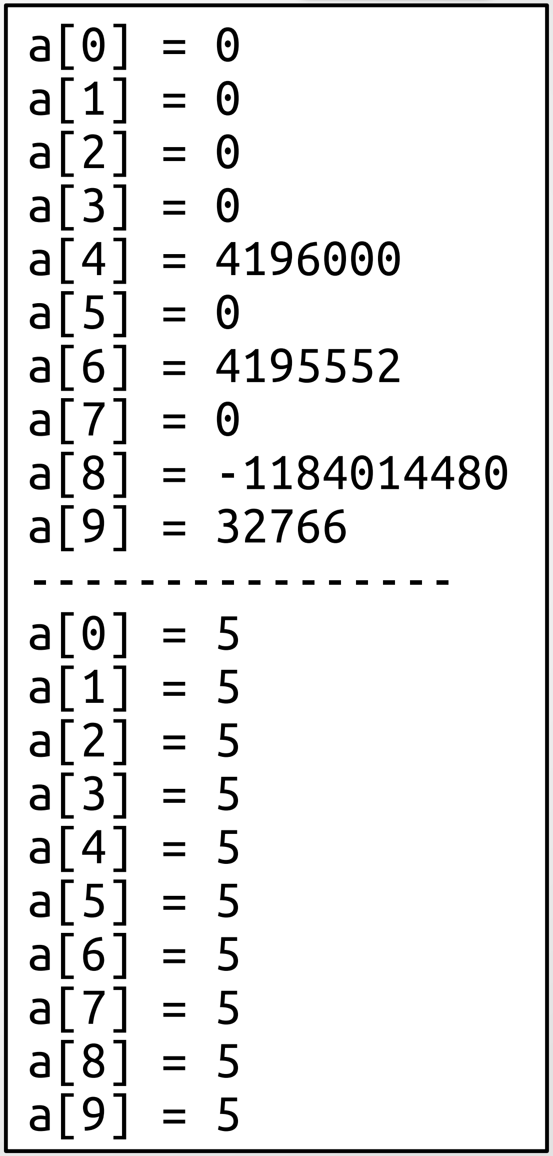
Arrays

- An array is a collection of values that have the same type.
- The size of an array is fixed (it cannot be changed).
- An array can be either one-dimensional or multidimensional.
- The most commonly used arrays are one-dimensional and two-dimensional.
- Values are selected by integer indexes.
- Indexes always start at 0.

Declaring Arrays

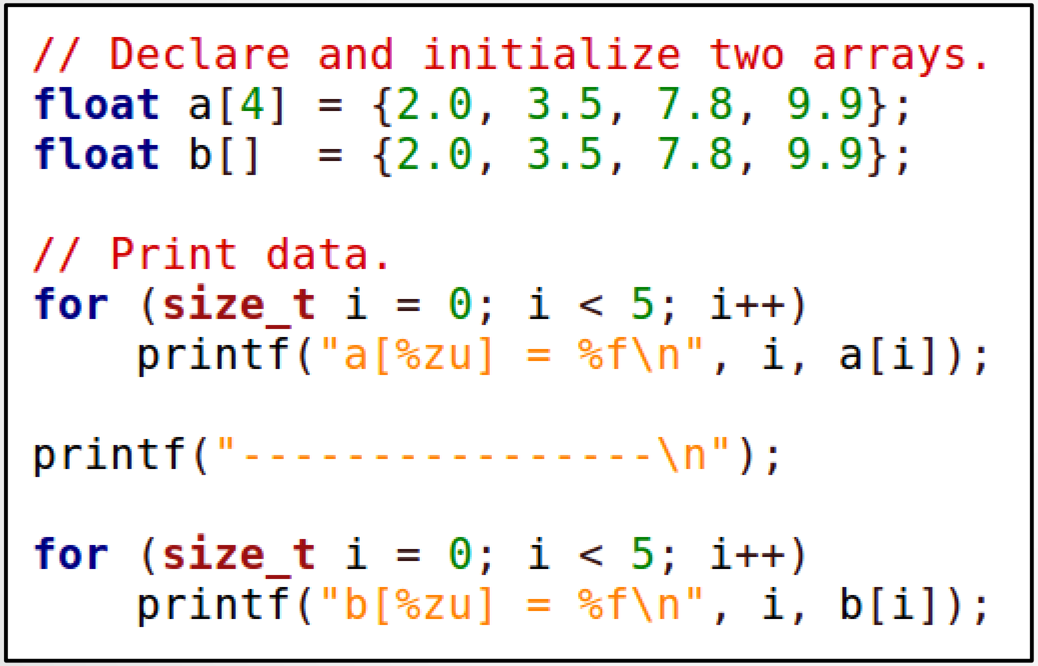


```c
#include <stdio.h>

#define SIZE 10

int main()
{
    // Declare an array (undefined data).
    int a[SIZE];

    // Print data (undefined).
    for (size_t i = 0; i < SIZE; i++)
        printf("a[%zu] = %i\n", i, a[i]);
    printf("----------------\n");

    // Initialize data.
    for (size_t i = 0; i < SIZE; i++)
        a[i] = 5;

    // Print data.
    for (size_t i = 0; i < SIZE; i++)
        printf("a[%zu] = %i\n", i, a[i]);
}
```

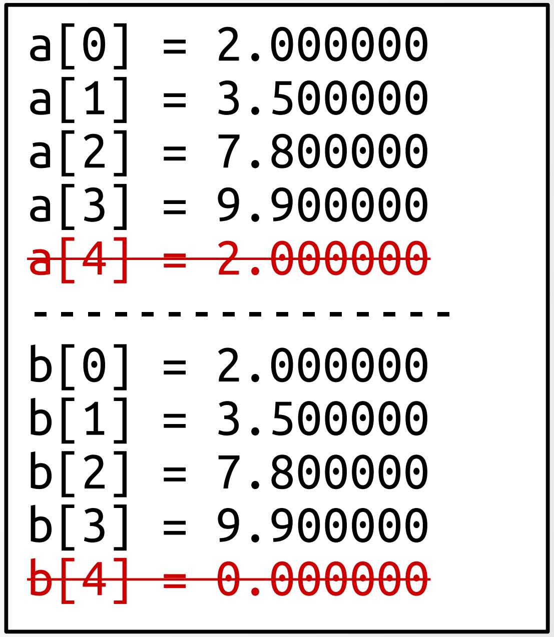


```
a[0] = 0
a[1] = 0
a[2] = 0
a[3] = 0
a[4] = 4196000
a[5] = 0
a[6] = 4195552
a[7] = 0
a[8] = -1184014480
a[9] = 32766
----------------
a[0] = 5
a[1] = 5
a[2] = 5
a[3] = 5
a[4] = 5
a[5] = 5
a[6] = 5
a[7] = 5
a[8] = 5
a[9] = 5
```

Declaring and Initializing Arrays



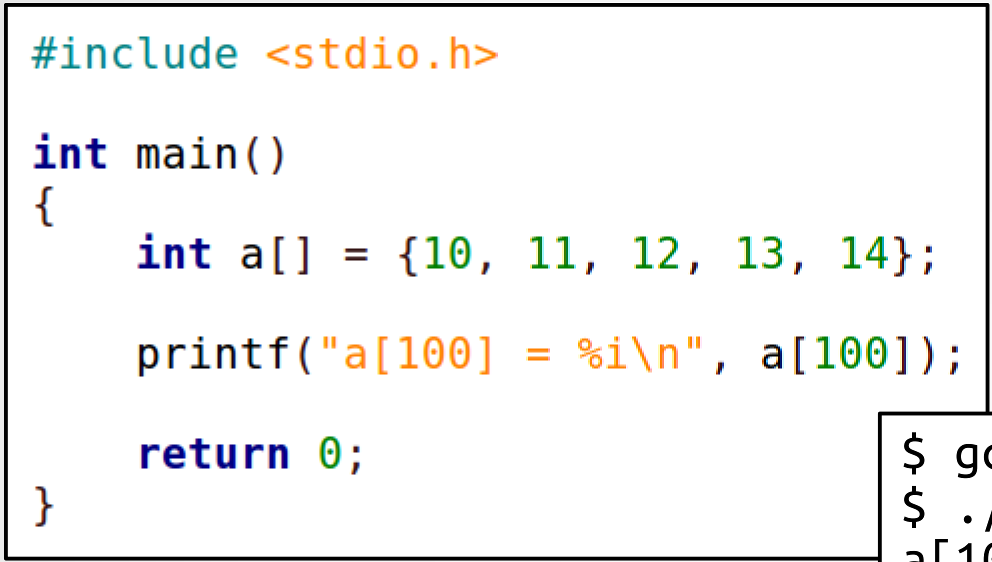
```c
// Declare and initialize two arrays.
float a[4] = {2.0, 3.5, 7.8, 9.9};
float b[]  = {2.0, 3.5, 7.8, 9.9};

// Print data.
for (size_t i = 0; i < 5; i++)
    printf("a[%zu] = %f\n", i, a[i]);

printf("----------------\n");

for (size_t i = 0; i < 5; i++)
    printf("b[%zu] = %f\n", i, b[i]);
```



```
a[0] = 2.000000
a[1] = 3.500000
a[2] = 7.800000
a[3] = 9.900000
~~a[4] = 2.000000~~
----------------
b[0] = 2.000000
b[1] = 3.500000
b[2] = 7.800000
b[3] = 9.900000
~~b[4] = 0.000000~~
```

Out of Bound Access – Reading

out_of_bound.c



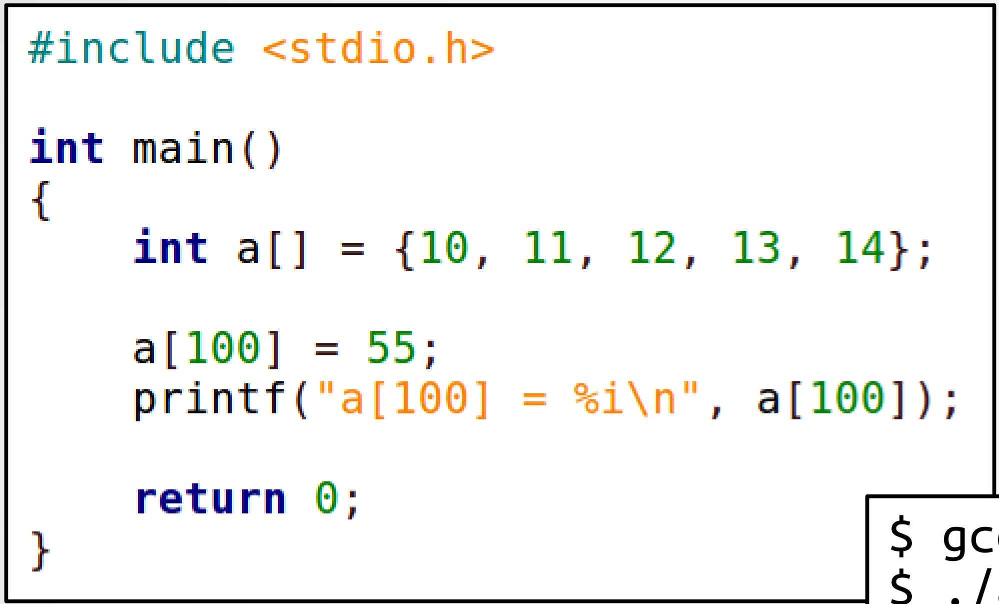
```c
#include <stdio.h>

int main()
{
    int a[] = {10, 11, 12, 13, 14};

    printf("a[100] = %i\n", a[100]);

    return 0;
}
```

```
$ gcc -Wall -Wextra out_of_bound.c
$ ./a.out
a[100] = -692211237
$ ./a.out
a[100] = -2024418853
$ ./a.out
a[100] = 1341063643
```

No compilation errors!
No compilation warnings!
Undefined behavior!

Out of Bound Access – Writing

out_of_bound.c

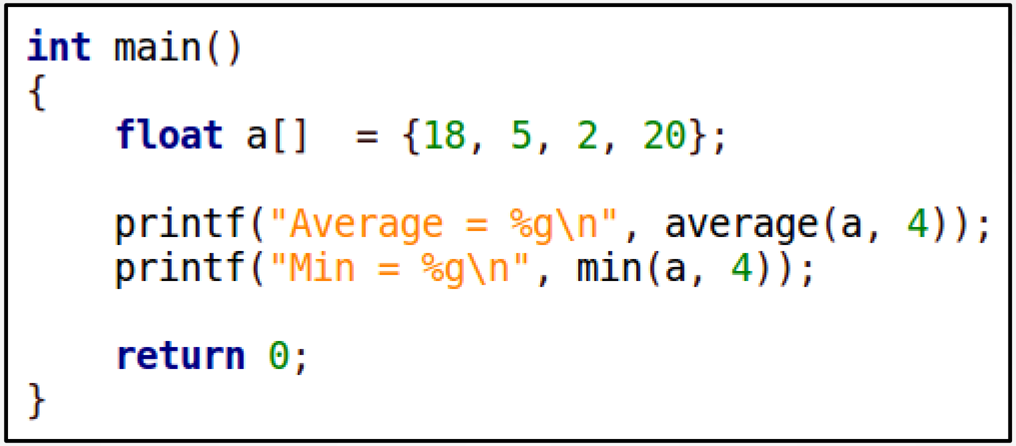


```c
#include <stdio.h>

int main()
{
    int a[] = {10, 11, 12, 13, 14};

    a[100] = 55;
    printf("a[100] = %i\n", a[100]);

    return 0;
}
```

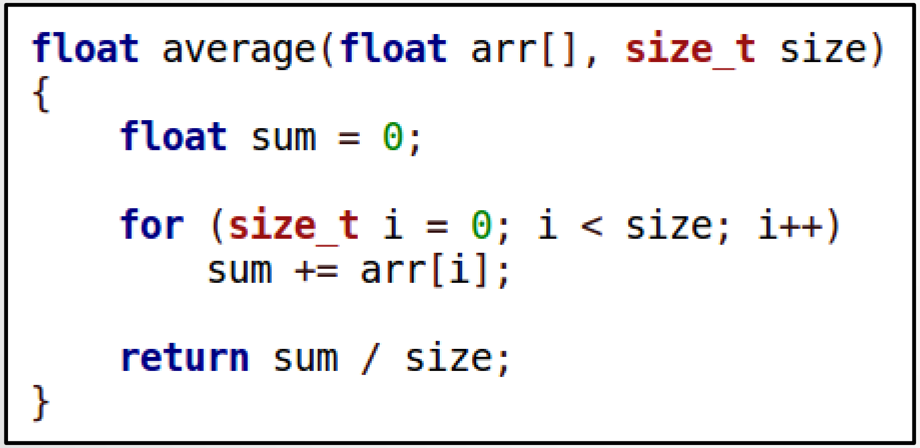
```
$ gcc -Wall -Wextra out_of_bound.c
$ ./a.out
Segmentation fault (core dumped)
```

No compilation errors!
No compilation warnings!
Segmentation fault (access violation)!
The program crashes!

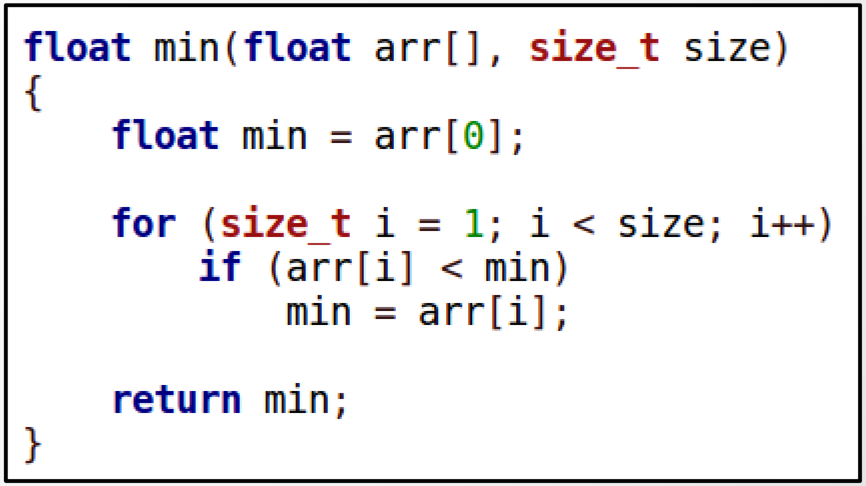
Manipulating Arrays – Example



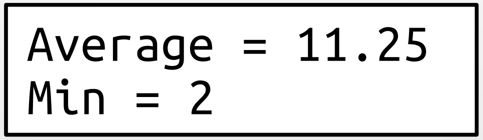
```c
int main()
{
    float a[]  = {18, 5, 2, 20};

    printf("Average = %g\n", average(a, 4));
    printf("Min = %g\n", min(a, 4));

    return 0;
}
```



```c
float average(float arr[], size_t size)
{
    float sum = 0;

    for (size_t i = 0; i < size; i++)
        sum += arr[i];

    return sum / size;
}
```

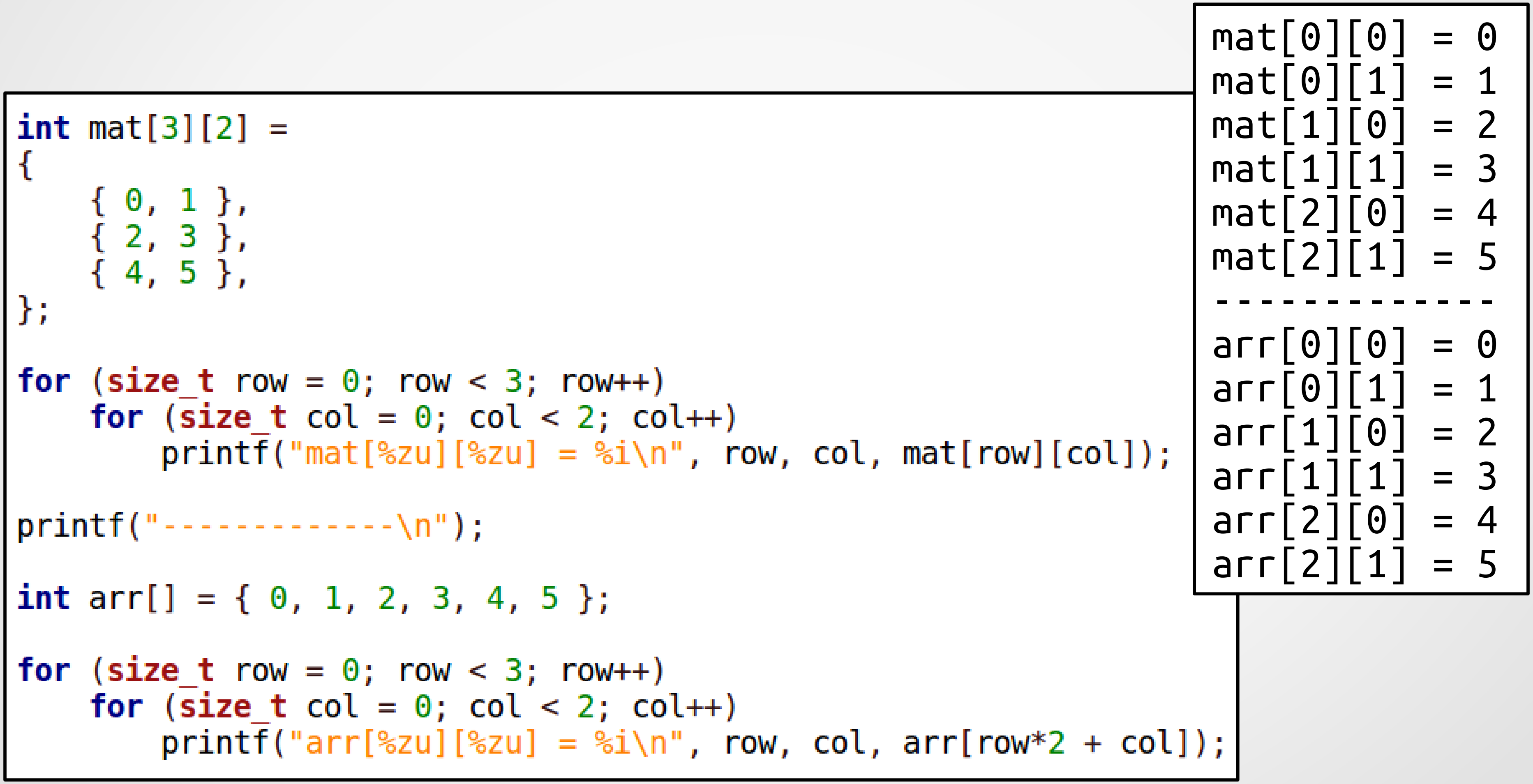


```c
float min(float arr[], size_t size)
{
    float min = arr[0];

    for (size_t i = 1; i < size; i++)
        if (arr[i] < min)
            min = arr[i];

    return min;
}
```



```
Average = 11.25
Min = 2
```

Two-Dimensional Arrays



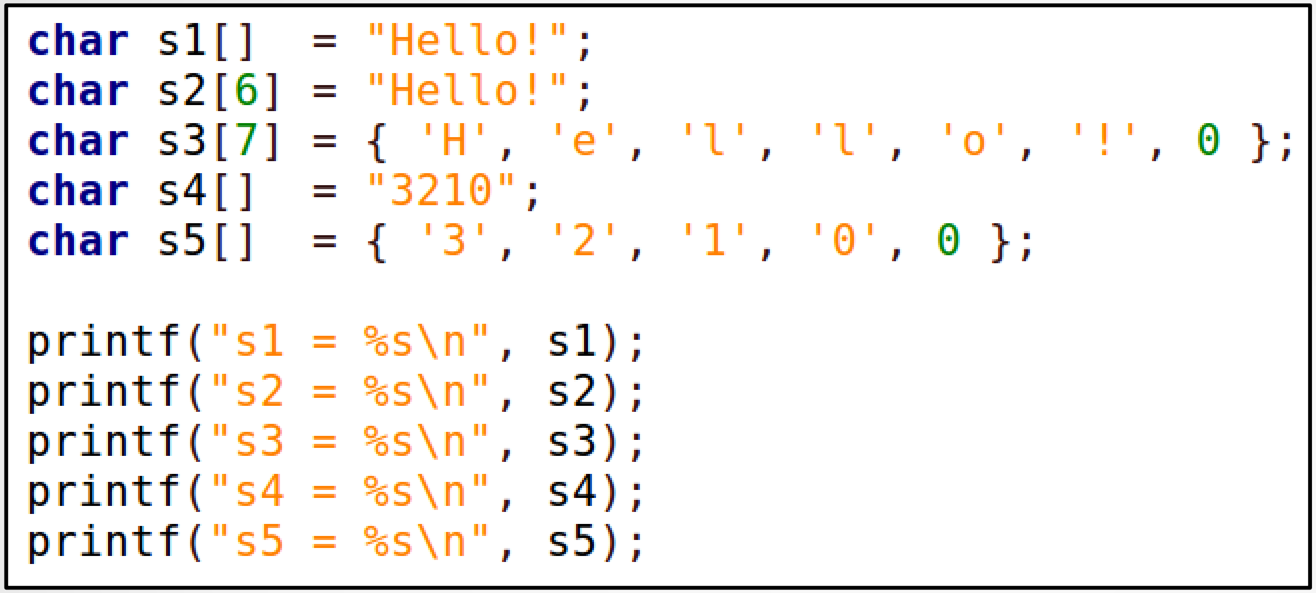
```
int mat[3][2] =
{
    { 0, 1 },
    { 2, 3 },
    { 4, 5 },
};

for (size_t row = 0; row < 3; row++)
    for (size_t col = 0; col < 2; col++)
        printf("mat[%zu][%zu] = %i\n", row, col, mat[row][col]);

printf("------------\n");

int arr[] = { 0, 1, 2, 3, 4, 5 };

for (size_t row = 0; row < 3; row++)
    for (size_t col = 0; col < 2; col++)
        printf("arr[%zu][%zu] = %i\n", row, col, arr[row*2 + col]);
```

```
mat[0][0] = 0
mat[0][1] = 1
mat[1][0] = 2
mat[1][1] = 3
mat[2][0] = 4
mat[2][1] = 5
-------------
arr[0][0] = 0
arr[0][1] = 1
arr[1][0] = 2
arr[1][1] = 3
arr[2][0] = 4
arr[2][1] = 5
```

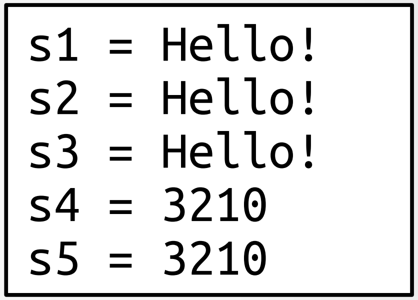
Strings of Characters

- A string is an array of characters.
- A string is always terminated by a null character (the ASCII code 0).
- Characters are selected by integer indexes.
- Indexes always start at 0.

Declaring Strings



```
char s1[]  = "Hello!";
char s2[6] = "Hello!";
char s3[7] = { 'H', 'e', 'l', 'l', 'o', '!', 0 };
char s4[]  = "3210";
char s5[]  = { '3', '2', '1', '0', 0 };

printf("s1 = %s\n", s1);
printf("s2 = %s\n", s2);
printf("s3 = %s\n", s3);
printf("s4 = %s\n", s4);
printf("s5 = %s\n", s5);
```



```
s1 = Hello!
s2 = Hello!
s3 = Hello!
s4 = 3210
s5 = 3210
```

s1, ***s2*** and ***s3*** are identical.
s4 and ***s5*** are identical.

Do not confuse **'0'** (ASCII code: 48) and
0 (the null character ; ASCII Code: 0).

See also: table of escape sequences.

Manipulating Strings – Example

```c
char s1[] = "Hello";
char s2[] = "World!";
char s3[] = "Bye";
char buffer[50];

size_t l1 = str_len(s1);
size_t l2 = str_len(s2);
size_t l3 = str_len(s3);

printf("l1 = %zu\n", l1);
printf("l2 = %zu\n", l2);
printf("l3 = %zu\n", l3);

str_cp(s1, l1, buffer, 50);
printf("buffer = %s\n", buffer);

str_cp(s2, l2, buffer, 50);
printf("buffer = %s\n", buffer);

str_cp(s3, l3, buffer, 50);
printf("buffer = %s\n", buffer);
```

```c
void str_cp(char src[], size_t src_len, char dst[], size_t dst_len)
{
    if (src_len >= dst_len)
        return;

    for (size_t i = 0; i <= src_len; i++)
        dst[i] = src[i];
}
```

```c
size_t str_len(char s[])
{
    size_t i = 0;

    while (s[i] != 0)
        i++;

    return i;
}
```

```
l1 = 5
l2 = 6
l3 = 3
buffer = Hello
buffer = World!
buffer = Bye
```

Command-Line Arguments

```
$ gcc -Wall -Wextra args.c
```

```
$ ./a.out
Number of arguments ......... 1
argv[0] (program name) ...... "./a.out"
```

```
$ ./a.out a bc "d e" " fg  " h
Number of arguments ......... 6
argv[0] (program name) ...... "./a.out"
argv[1] ..................... "a"
argv[2] ..................... "bc"
argv[3] ..................... "d e"
argv[4] ..................... " fg  "
argv[5] ..................... "h"
```

args.c

```c
#include <stdio.h>

int main(int argc, char** argv)
{
    printf("Number of arguments ......... %i\n", argc);

    printf("argv[0] (program name) ...... \"%s\"\n", argv[0]);
    for (int i = 1; i < argc; i++)
        printf("argv[%i] .................... \"%s\"\n", i, argv[i]);
}
```