cc-practice-array

Interactive Notebook for Introduction to Programming in C/C++, CSC 100, Spring 2021

Table of Contents

1. One-dimensional arrays

1.1. TODO Declare and initialize array

1.1.1. Problem

1. Write two programs that declare an integer array `foo` and a floating-point array `bar`. There are **four** ways that you know already:
   - declare and initialize as `{...}` list
   - initialize individual elements
   - initialize using a loop (with `scanf`)
2. Initialize `foo` with the values `1,2,3,4,5`. Print the first and the last value of `foo`. Use the **four** different ways of initializing `foo` (don't create a new named code block for each way but create arrays `foo1`, `foo2` and `foo3` instead).
3. Initialize `bar` with the values `1.1,2.2,3.3,4.4,5.5`. Print the first and the last value of `bar`. Use the **four** different ways of initializing `bar` (don't create a new named code block for each way but create arrays `bar1`, `bar2` and `bar3` instead).
4. When you're done with `foo`, copy the code block and use `M-x replace-string` to replace `foo` by `bar` in the code block only (select the region first with `C-SPC`).

5. Sample output:

```
: foo1[0] = 1,       foo2[0] = 1,    foo3[0] = 1,
: foo1[1] = 2,       foo2[1] = 2,    foo3[1] = 2,
: foo1[2] = 3,       foo2[2] = 3,    foo3[2] = 3,
: foo1[3] = 4,       foo2[3] = 4,    foo3[3] = 4,
: foo1[4] = 5,       foo2[4] = 5,    foo3[4] = 5,
```

```
: bar1[0] = 1.1,     bar2[0] = 1.1,  bar3[0] = 1.1,
: bar1[1] = 2.2,     bar2[1] = 2.2,  bar3[1] = 2.2,
: bar1[2] = 3.3,     bar2[2] = 3.3,  bar3[2] = 3.3,
: bar1[3] = 4.4,     bar2[3] = 4.4,  bar3[3] = 4.4,
: bar1[4] = 5.5,     bar2[4] = 5.5,  bar3[4] = 5.5,
```

1.1.2. Sample solution

```
echo "1 2 3 4 5" > ./src/fooInput
cat ./src/fooInput
```

```
//declare foo2, foo3
int foo2[5], foo3[5];

// foo1 is initialized as list
int foo1[5] = {1,2,3,4,5};

// foo2 is initialized using individual elements
foo2[0] = 1;
foo2[1] = 2;
foo2[2] = 3;
foo2[3] = 4;
foo2[4] = 5;

// foo3 and foo4 are initialized with a loop
for (int i = 0; i < 5; i++) {
  scanf("%d", &foo3[i]);
  printf("foo1[%d] = %d,\tfoo2[%d] = %d,\tfoo3[%d] = %d\n",
         i, foo1[i], i, foo2[i], i, foo3[i]);
 }
```

```
foo1[0] = 1,    foo2[0] = 1,    foo3[0] = 1
foo1[1] = 2,    foo2[1] = 2,    foo3[1] = 2
foo1[2] = 3,    foo2[2] = 3,    foo3[2] = 3
foo1[3] = 4,    foo2[3] = 4,    foo3[3] = 4
foo1[4] = 5,    foo2[4] = 5,    foo3[4] = 5
```

```
echo "1.1 2.2 3.3 4.4 5.5" > ./src/barInput
cat ./src/barInput
```

```
float bar2[5], bar3[5];

// initialized as list
float bar1[5] = {1.1f,2.2f,3.3f,4.4f,5.5f};

// initialized individual elements
bar2[0] = 1.1f;
bar2[1] = 2.2f;
bar2[2] = 3.3f;
bar2[3] = 4.4f;
bar2[4] = 5.5f;

// loop initialization
for (int i = 0; i < 5; i++) {
  scanf("%f", &bar3[i]);
  printf("bar1[%d] = %g,\tbar2[%d] = %g,\tbar3[%d] = %.1f,\n",
         i, bar1[i], i, bar2[i], i, bar3[i]);
 }
```

```
bar1[0] = 1.1,  bar2[0] = 1.1,  bar3[0] = 1.1,
bar1[1] = 2.2,  bar2[1] = 2.2,  bar3[1] = 2.2,
bar1[2] = 3.3,  bar2[2] = 3.3,  bar3[2] = 3.3,
bar1[3] = 4.4,  bar2[3] = 4.4,  bar3[3] = 4.4,
bar1[4] = 5.5,  bar2[4] = 5.5,  bar3[4] = 5.5,
```

```
float bar2[5], bar3[5];

// initialized as list
float bar1[5] = {1.1f,2.2f,3.3f,4.4f,5.5f};

// initialized individual elements
bar2[0] = 1.1f;
bar2[1] = 2.2f;
bar2[2] = 3.3f;
bar2[3] = 4.4f;
bar2[4] = 5.5f;

// loop initialization
for (int i = 0; i < 5; i++) {
  bar3[i] = (11.f * i + 11.f) / 10.f;
  printf("bar1[%d] = %g,\tbar2[%d] = %g,\tbar3[%d] = %g,\n",
         i, bar1[i], i, bar2[i], i, bar3[i]);
 }
```

```
bar1[0] = 1.1,        bar2[0] = 1.1,  bar3[0] = 1.1,
bar1[1] = 2.2,        bar2[1] = 2.2,  bar3[1] = 2.2,
bar1[2] = 3.3,        bar2[2] = 3.3,  bar3[2] = 3.3,
bar1[3] = 4.4,        bar2[3] = 4.4,  bar3[3] = 4.4,
bar1[4] = 5.5,        bar2[4] = 5.5,  bar3[4] = 5.5,
```

1.2. TODO Sample program: reversing numbers

1.2.1. Problem

- Enter five numbers and print them in reverse order.
- Store the numbers as one array.
- Use the tips below to complete the code.

1.2.2. Solution

1. Input file - check file location

   ```
   echo '34 82 49 102 7' > ./src/numbers
   cat ./src/numbers
   ```

2. Code

   Fill in a few empty statements below:

   - At the top, define a macro `N` with the value `5`
   - Declare an integer array `a` of length `N`
   - Complete the `scanf` function inside the `for` loop to input values and to print values that were entered
   - Complete the second loop to print numbers in reverse

   - The output should look like this:

     ```
     : Enter 5 numbers: 34 82 49 102 7
     : In reverse order: 7 102 49 82 34
     ```

```
???   // define macro on this line (upper bound)

???   // declare array a[N] on this line

int i; // counting variable ( can also define inside for loop)

printf("Enter %d numbers: ", N);

for ( ... ) {
  ...   // input array values
  ...   // print each array value
 }

printf("In reverse order:");
for ( ... ) {
  printf(" %d", a[i]);
 }
printf("\n");
```

3. Sample solution

```
#define N 5  // define macro on this line (upper bound)

int a[N];   // declare array a[N] on this line
int i; // counting variable ( can also define inside for loop)

printf("Enter %d numbers: ", N);

for ( i = 0; i < N; i++ ) {
  scanf("%d", &a[i]);   // input array values
  printf("%d ", a[i]);   // print each array value
 }

printf("\nIn reverse order:");
for ( i = N-1; i >= 0; i--) {
  printf(" %d", a[i]);
 }
printf("\n");
```

```
Enter 5 numbers: 34 82 49 102 7
In reverse order: 7 102 49 82 34
```

2. Multi-dimensional arrays

2.1. TODO Print a 2-dimensional array

- Declare a 4 x 3 matrix and print it in 2 dimensions. Complete the sample code below to get the output shown.
- Initialize the matrix `m[][]` with the elements 1 through 12.
- Define two nested `for` loops looping over rows and columns

- Output:

```
#+RESULTS:
:  1  2  3
:  4  5  6
:  7  8  9
: 10 11 12
```

- Code:

```
... // declare and initialize matrix

for (...) {  // loop over rows
  for(...) {  // loop over columns
    ... // print matrix elements
      }
  printf("\n");
 }
```

1. Sample solution

```
int m[4][3] = {1,2,3,4,5,6,7,8,9,10,11,12};

for (int i=0;i<4;i++) {
  for(int j=0;j<3;j++) {
    printf("%3d", m[i][j]);
  }
  printf("\n");
 }
```

```
 1  2  3
 4  5  6
 7  8  9
10 11 12
```

```
 1  2  3
 4  5  6
 7  8  9
10 11 12
```

2.2. TODO Use `sizeof` in a `for` loop

- The code block below defines an array `a` of length `5`. Complete the `for` loop using the `sizeof` operator to get the output shown. The loop re-initializes the array `a`.

- Output:

```
#+RESULTS:
: a[0] = 1
: a[1] = 1
: a[2] = 1
: a[3] = 1
: a[4] = 1
```

- Run the code, then change the length of the array and re-run the code.

- Code:

```
int a[5] = {0}; // initialize array

for (int i = 0; i < sizeof(a)/sizeof(a[0]); i++) {
  a[i] = 1; // re-initialize array
  printf("a[%d] = %d\n", i, a[i]);
 }
```

```
a[0] = 1
a[1] = 1
a[2] = 1
a[3] = 1
a[4] = 1
```

Author: Marcus Birkenkrahe [pledged]
Created: 2022-06-20 Mon 21:41