8 Concatenating Two Arrays Using VLA Parameters

CSC 240 (Data Structures) Lyon College Fall 2024

Marcus Birkenkrahe (pledged)

October 24, 2024

Problem Statement

In this assignment, you will define a function to concatenate two arrays `a` and `b` into a third array `c` using Variable-Length Array (VLA) parameters. The sizes of `a` and `b` are determined at runtime, and the function should copy both arrays successively into `c`.

Tasks

1. Define a function `concat` that takes two arrays `a` and `b`, and a result array `c`. The function should concatenate the two arrays into the third one.

   Function prototype:

   ```
   void concat(int m, int n, int a[m], int b[n], int c[m+n]);
   ```

2. Implement a simple example that concatenates two arrays `a = {1,2,3}` and `b = {4,5}`, calls the `concat` function, and prints the resulting concatenated array `c`.

3. Bonus: Create another version of the program that calls the function using compound literals.

Expected Output

```
Array a: 1, 2, 3
Array b: 4, 5
Concatenated array c: 1 2 3 4 5
```

Submission

1. **Meta data header** (title, author/pledge, subtitle, startup, property).
2. **Headlines** to structure your program (Problem, Implementation, Reflection).
3. **Code block(s)** with program header, function prototype and function documentation (purpose).
4. **Short reflection** about your experience and learning.

Solution

Meta Data Header

```
#+title: Concatenating Two Arrays Using VLA Parameters
#+author: Marcus Birkenkrahe (pledged)
#+subtitle: CSC 240 Data Structures, Lyon College, Fall 2024
#+startup: overview hideblocks indent
#+property: header-args:C :main yes :includes <stdio.h> :results output
```

Problem

1. Declare three VLAs a,b,c.
2. Initialize two arrays a,b.
3. Call function `concat` that concatenates a and b into c.
4. Print resulting array c.
5. Create another version where `concat` is called on two *compound literals* to be concatenated.

Implementation

- Our approach will be to move as much of the action into functions.
- Pseudocode:

```
// function: get array lengths n,m
// function: get array elements
// function: print array
// function: concatenate two VLAs a,b to c
// main pgm
   // define array lengths n,m
   // get array lengths n,m
   // define VLAs a[n], b[m], c[n+m]
   // get array elements
   // print arrays a and b
   // concatenate arrays a and b to c
   // print array c
```

- Code: Version 1 with static array lengths defined at the start

```
#define N 3
#define M 2

// function prototypes:
// function: get array elements
void get_array(int [], int);

// function: print array
void print_array(int [], int);

// function: concatenate two VLAs a,b to c
void concat(int n, int m, int a[n], int b[m], int c[n+m]);

// main pgm
int main(void) {

  // define array lengths n,m
  int n=N, m=M;

  // define VLAs a[n], b[m], c[n+m]
```

```
  int a[n], b[m], c[n+m];

  // get array elements
  get_array(a,n);
  get_array(b,m);

  // print arrays a and b
  print_array(a,n);
  print_array(b,m);

  // concatenate arrays a and b to c
  concat(n, m, a, b, c);

  // print array c
  print_array(c, n+m);

  return 0;
}
// function definitions
void get_array(int array[], int length) {
  for (int i=0;i<length; i++)
    scanf("%d", &array[i]);
}

void print_array(int array[], int length) {
  for (int i=0;i<length; i++)
    printf("%d", array[i]);
  puts("");
}

void concat(int n, int m, int a[n], int b[m], int c[n+m]) {
  for (int i=0;i<n; i++)
    c[i] = a[i];
  for (int i=0;i<m; i++)
    c[i+n] = a[i];
}
```

- Testing suite:

```
gcc concat.c -o concat
```

```
echo 1 2 3 4 5 | ./concat
```

- Code: Version 2 - all data as user input

```
/************************************************/
// concat2.c: concatenate VLAs a,b to array c
/************************************************/
// function prototypes:
// function get array lengths
void get_array_lengths(int*, int*);

// function: get array elements
void get_array(int [], int);

// function: print array
void print_array(int [], int);

// function: concatenate two VLAs a,b to c
void concat(int n, int m, int a[n], int b[m], int c[n+m]);

// main pgm
int main(void) {

  // define array lengths n,m
  int n, m;

  // get array lengths
  get_array_lengths(&n, &m);

  // define VLAs a[n], b[m], c[n+m]
  int a[n], b[m], c[n+m];

  // get array elements
  get_array(a,n);
  get_array(b,m);

  // print arrays a and b
  printf("Array a: ");
  print_array(a,n);
  printf("Array b: ");
```

```
  print_array(b,m);

  // concatenate arrays a and b to c
  concat(n, m, a, b, c);

  // print array c
  printf("Concatenated array c: ");
  print_array(c, n+m);

  return 0;
}
// function definitions
void get_array_lengths(int* length_a, int* length_b) {
  scanf("%d%d", length_a, length_b);
  }

void get_array(int array[], int length) {
  for (int i=0;i<length; i++)
    scanf("%d", &array[i]);
}

void print_array(int array[], int length) {
  for (int i=0;i<length; i++)
    printf("%d ", array[i]);
  puts("");
}

void concat(int n, int m, int a[n], int b[m], int c[n+m]) {
  for (int i=0;i<n; i++)
    c[i] = a[i];
  for (int i=0;i<m; i++)
    c[i+n] = b[i];
}
```

- Testing suite:

```
gcc concat2.c -o concat2
echo 3 2 1 2 3 4 5 | ./concat2  # n m a[0] a[1] a[2] b[0] b[1]
```

Bonus: Compound literals

- Code: Version 3 - call `concat` on two compound literals

```
/************************************************/
// concat3.c: concatenate arrays a,b to array c
// Version 3: arrays initialized as compound literals
/************************************************/

// function: concatenate two VLAs a,b to c
void concat(int n, int m, int a[n], int b[m], int c[n+m]);

// main pgm
int main(void) {

  // define array lengths n,m
  int n=3, m=2, c[n+m];

  // concatenate arrays a and b to c
  printf("Concatenated array c: \n");
  concat(n, m, (int []){1,2,3}, (int []){4,5}, c);

  return 0;
}

void concat(int n, int m, int a[n], int b[m], int c[n+m]) {
  for (int i=0;i<n; i++) {
    c[i] = a[i];
    printf("%d ", c[i]);
  }
  for (int i=0;i<m; i++) {
    c[i+n] = b[i];
    printf("%d ", c[i+n]);
  }
}


Concatenated array c:
1 2 3 4 5
```