Laboratory 11 – Strings and 2D arrays

21.12.2021

1. [5points] There are two strings `str1` and `str2`. Write a function `swapstr` which copies the contents of `str1` to `str2`, and the contents of `str2` to `str1`.

A. In the `main` function:

a) Declare two strings and set them to values "`abcdefg`" and "`ijklmno`" respectively. Don't use `malloc`.

b) Print both strings.

c) Call the `swapstr`. The first argument is the address of the first string, the second argument is the address of the second string, and the third argument is the size of the first string. The function returns nothing. Calculate the size of the string using the `strlen` function.

b) Print both strings.

B. Definition of the `swapstr` function.

a) With the `malloc` function create a temporary variable into which you copy the contents of one of the strings.

b) If the allocation was successful, call `strncpy` 3 times to swap the contents of the strings. Free up memory by calling the `free` function.

C. If possible, use `valgrind` to check memory usage.

Test data:

```
str1 = abcdefg, str2 = ijklmno
str1 = ijklmno, str2 = abcdefg
```

C. `valgrind ./a.out`

```
==136727== Memcheck, a memory error detector
==136727== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==136727== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==136727== Command: ./a.out
==136727==
str1 = abcdefg, str2 = ijklmno
str1 = ijklmno, str2 = abcdefg
==136727==
==136727== HEAP SUMMARY:
==136727==     in use at exit: 0 bytes in 0 blocks
==136727==   total heap usage: 2 allocs, 2 frees, 1,031 bytes allocated
==136727==
==136727== All heap blocks were freed -- no leaks are possible
==136727==
==136727== For lists of detected and suppressed errors, rerun with: -s
==136727== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

2. [2+1+3+4 points] In the `main` function, declare 3 two-dimensional arrays of the `float` type. The size of the array is defined with `#define SIZE 3.`

All functions are of the `void` type.

a. [2points] Write a function that assigns pseudo-random values to all the elements of the two-dimensional array. In the `main` function: Call a function for two previously declared arrays.

b. [1points] Write a function that prints all the elements of the two-dimensional array.

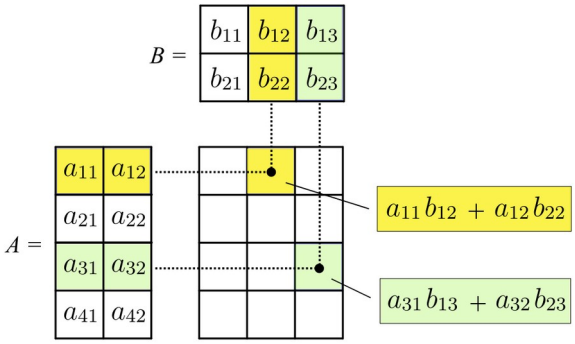
In the `main` function: Call a function for two previously initialized arrays.

c. [3points] Write a function that will calculate the result of adding two arrays and store the result in a third.

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix} + \begin{bmatrix} b00 & b01 & b02 \\ b10 & b11 & b12 \\ b20 & b21 & b22 \end{bmatrix} = \begin{bmatrix} a00+b00 & a01+b01 & a02+b02 \\ a10+b10 & a11+b11 & a12+b12 \\ a20+b20 & a21+b21 & a22+b22 \end{bmatrix} = \begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix}$$

In the `main` function: Call the function for two previously initialized arrays and print the result.

d. [4points] Write a function that will calculate the result of the multiplication of two arrays and store the result in the third.



If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is an $n \times p$ matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{pmatrix}$$

the *matrix product* $\mathbf{C} = \mathbf{AB}$ (denoted without multiplication signs or dots) is defined to be the $m \times p$ matrix[6][7][8][9]

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{pmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj},$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, p$.

That is, the entry c_{ij} of the product is obtained by multiplying term-by-term the entries of the ith row of $\mathbf{A}$ and the jth column of $\mathbf{B}$, and summing these n products. In other words, c_{ij} is the dot product of the ith row of $\mathbf{A}$ and the jth column of $\mathbf{B}$.[1]

Therefore, $\mathbf{AB}$ can also be written as

$$\mathbf{C} = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & a_{11}b_{12} + \cdots + a_{1n}b_{n2} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ a_{21}b_{11} + \cdots + a_{2n}b_{n1} & a_{21}b_{12} + \cdots + a_{2n}b_{n2} & \cdots & a_{21}b_{1p} + \cdots + a_{2n}b_{np} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & a_{m1}b_{12} + \cdots + a_{mn}b_{n2} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Thus the product $\mathbf{AB}$ is defined if and only if the number of columns in $\mathbf{A}$ equals the number of rows in $\mathbf{B}$,[2] in this case n.

https://en.wikipedia.org/wiki/Matrix_multiplication

In the `main` function: Call the function for two previously initialized arrays and print the result.

`Test data:`

```
a:
0.241978 0.511284 0.570594
0.032029 0.223468 0.288882
0.859432 0.327955 0.125259
b:
0.602074 0.546275 0.783408
0.890478 0.631554 0.428266
0.488831 0.020935 0.669333

Sum of matrices a + b
0.844053 1.057559 1.354002
0.922507 0.855022 0.717148
1.348263 0.348890 0.794591

Product of matrices a * b
0.879900 0.467036 0.790451
0.359491 0.164676 0.314153
0.870709 0.679229 0.897578
```

3. [2+3points] Pointers and 2D arrays.

a. [2points] Write a function that computes the sum of the values stored in each row in a two-dimensional array and stores the result in a one-dimensional array. Do not use the [] operator, use the appropriate pointer.

b. [3points] Write a function that computes the sum of the values stored in each column in a two-dimensional array and stores the result in a one-dimensional array. Do not use the [] operator, use the appropriate pointer.

In the `main` function, declare and initialize the appropriate arrays, then call the defined functions and print the results.

`Test data:`

```
A:
0.840188 0.394383 0.783099
0.798440 0.911647 0.197551
0.335223 0.768230 0.277775

Sum of elements of Row 0 = 2.017670
Sum of elements of Row 1 = 1.907639
Sum of elements of Row 2 = 1.381227

Sum of elements of Col 0 = 1.973850
Sum of elements of Col 1 = 2.074260
Sum of elements of Col 2 = 1.258425
```

Next lab 12 – Structures.