Laboratory 14 – pointers to functions, qsort 19.01.2021

1.[5points] Write a program that allocates memory for an array of size specified by the user at runtime. Assign pseudo-random values to the elements of the array, then sort the array using the `qsort` function.

A. In the `main` function:

a) Ask the user for the size of the array.

b) Using the `malloc` function allocate the `double` array of the size specified by the user.

c) Check if the allocation was successful.

- If the address returned by `malloc` is not `NULL`, use the `rand` function in a `for` loop and assign pseudo random values to the elements of the array. Print the array. Sort the array by calling `qsort`. Print the array. Free memory by calling the `free` function.

- If the allocation failed and the address returned by `malloc` is `NULL`, print the message and exit the program.

B. Define a function that compares elements of type `double`. The function is similar to the example in lecture 13.

`Test data:`

```
Enter the size of the array.
12
Before qsort:
0.897192 0.832734 0.153990 0.301569 0.532044 0.322341 0.311172 0.684078 0.238005 0.027206 0.819926 0.423477
After qsort:
0.027206 0.153990 0.238005 0.301569 0.311172 0.322341 0.423477 0.532044 0.684078 0.819926 0.832734 0.897192
```

2. [5points] Modify the source code of the program from Exercise 1 to sort the string (character array).

a) When allocating memory to an array, be sure to add “\0” at the end of the array.

b) Assign random letters from ‘a’ to ‘z’ to the array elements.

c) Print the array of characters using the `printf("%s\n", arr);` command.

d) When using the `qsort` function, remember that ‘\0’ must remain at the end of the array.

`Test data:`

```
Enter the size of the array.
50
Before qsort:
cukhvodptemwcwsxfavavdblgmnimiopezwbpbqihfhjbzjgcg
After qsort:
aabbbbccddeeffgggghhhiiijjklmmmnoopppqstuvvvwwwxzz
```

3. [10points] List of student names - write a program that allocates an array of strings.

A. In the `main` function:

a) Declare a pointer to a pointer to char (`namesList`). Set it to NULL.

b) Declare a variable (`nmbr`) whose value equals the number of strings in the array. Set it to `0`.

c) We add the strings to the array as follows:

```
namesList = AddStudent(namesList, "Chuck", &nmbr);
namesList = AddStudent(namesList, "Joe", &nmbr);
namesList = AddStudent(namesList, "Ann", &nmbr);
```

d) Call a function that will print the contents of the array of strings. Pass two variables `namesList` and `nmbr` to the function.

e) Call a function that frees memory. Pass two variables `namesList` and `nmbr` to the function.

B. Define the `AddStudent` function.

a) Using the variables passed to the function, allocate an array of pointers to `char`. We allocate the array using the `realloc` function, increasing the size of the array by one with each function call.

b) If the `realloc` function worked correctly, we add the student's name to the array .

c) To add the student's name to the array we need to use the `malloc` function to allocate the string, where we will copy the student's name. Use `strlen` to determine the size of the string. Don't forget about '\0'. Assign the result of the `malloc` function to the string array cell.

d) If the `malloc` function worked correctly, `strcpy` can be used to copy the student's name to the array of strings.

e) Increase the value of the variable that holds the size of the string array.

f) Write code that must be executed when `realloc` or `malloc` fails.

g) Complete the statement `return` .....`;`

C. Define the function that will print the student list. The function takes two arguments: pointer to pointer to char and an integer. Print the strings in one `for` loop.

D. Define the function that frees memory. The function takes two arguments: pointer to pointer to char and an integer.

a) In for loop, free memory where student names were stored.

b) Outside the loop, free memory for the string array.

c) We don't need dangling pointers. Complete the statement `return` .....`;`

Test data:

```
We have 3 students.
Students:
[#1 student in the list]:: Chuck
[#2 student in the list]:: Joe
[#3 student in the list]:: Ann
```

If you have valgrind (https://valgrind.org/) installed on your computer, you can check how the program handles memory allocation and deallocation.

A. Report for exercise 1.

`valgrind ./a.out`

```
==16300== Memcheck, a memory error detector
==16300== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==16300== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==16300== Command: ./a.out
==16300==
Enter the size of the array.
5
Before qsort:
0.527733 0.924733 0.275121 0.210389 0.567771
After qsort:
0.210389 0.275121 0.527733 0.567771 0.924733
==16300==
==16300== HEAP SUMMARY:
==16300==     in use at exit: 0 bytes in 0 blocks
==16300==   total heap usage: 3 allocs, 3 frees, 2,088 bytes allocated
==16300==
==16300== All heap blocks were freed -- no leaks are possible
==16300==
==16300== For lists of detected and suppressed errors, rerun with: -s
==16300== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

B. Report for exercise 2.

`valgrind ./a.out`

```
==31691== Memcheck, a memory error detector
==31691== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==31691== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==31691== Command: ./a.out
==31691==
Enter the size of the array.
50
Before qsort:
ijdbdyypezxwnnivntskvupxygrbwsjgbmhgnfytgvrukcpzvj
After qsort:
bbbcddefgggghiijjjkkmnnnnppprrssttuuvvvvwwxxyyyyzz
==31691==
==31691== HEAP SUMMARY:
==31691==     in use at exit: 0 bytes in 0 blocks
==31691==   total heap usage: 3 allocs, 3 frees, 2,099 bytes allocated
==31691==
==31691== All heap blocks were freed -- no leaks are possible
==31691==
==31691== For lists of detected and suppressed errors, rerun with: -s
==31691== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

C. Report for exercise 3.

`valgrind ./a.out`

```
==39741== Memcheck, a memory error detector
==39741== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==39741== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==39741== Command: ./a.out
==39741==
We have 3 students.
Students:
[#1 student in the list]:: Chuck
[#2 student in the list]:: Joe
[#3 student in the list]:: Ann
==39741==
==39741== HEAP SUMMARY:
==39741==     in use at exit: 0 bytes in 0 blocks
==39741==   total heap usage: 7 allocs, 7 frees, 1,086 bytes allocated
==39741==
==39741== All heap blocks were freed -- no leaks are possible
==39741==
==39741== For lists of detected and suppressed errors, rerun with: -s
==39741== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Next lab 15 – bsearch, files