세계 속의 빛 광주과학기술원

Computer Programming

Lab11

June 2, 2025

Submission

- **Submit to server**

Lab #

Class #

At the end of the Lab11, submit your C sources file by typing

~gs1401/bin/submit Lab11_2 ex11_1.c ex11_2.c ex11_3.c ex11_extra.c // by Thur. 11:50

~gs1401/bin/submit Lab11_3 ex11_1.c ex11_2.c ex11_3.c ex11_extra.c // by Friday 10:50

~gs1401/bin/submit Lab11_4 ex11_1.c ex11_2.c ex11_3.c ex11_extra.c // by Friday 11:50

~gs1401/bin/submit Lab11_5 ex11_1.c ex11_2.c ex11_3.c ex11_extra.c // by Friday 13:50

You may check that you have submitted your source code correctly by typing

~gs1401/bin/submit -check

Ex1

- Write a program that reads two integers *a* and *b*, then swaps their values using a function named swap(). The function must use pointers as parameters and perform the value exchange inside the function. The program should print the values of *a* and *b* both before and after the swap.
  1. void swap(int *x, int *y);
  2. The function should swap the values of the two integers using pointer dereferencing.
  3. Do not perform any value swapping in main(). The swapping must happen inside the swap() function.

Ex1

- **Program output**

```
[ohyong@cse ~/cp/Lab11]$ vi ex11_1.c
[ohyong@cse ~/cp/Lab11]$ gcc ex11_1.c -o ex11_1
[ohyong@cse ~/cp/Lab11]$ ./ex11_1
Enter two integers:
3 7
Before: a = 3, b = 7
After: a = 7, b = 3
```

```
[ohyong@cse ~/cp/Lab11]$ ./ex11_1
Enter two integers:
5 10
Before: a = 5, b = 10
After: a = 10, b = 5
```

Ex2

- Write a program that reads an integer *n* from the user, dynamically allocates memory for an array of *n* integers, takes *n* integers as input, computes their sum, and prints the result.
  - Use malloc() for memory allocation.
  - Include #include <stdlib.h> for using malloc() and free().
  - Do not use global variables.
  - Make sure to free the allocated memory using free().
  - Handle the case where malloc() returns NULL.
    - ✓ if (arr == NULL)
      return 1;

Ex2

- **Program output**

```
[ohyong@cse ~/cp/Lab11]$ vi ex11_2.c
[ohyong@cse ~/cp/Lab11]$ gcc ex11_2.c -o ex11_2
[ohyong@cse ~/cp/Lab11]$ ./ex11_2
Enter number of elements: 5
Enter 5 integers: 1 2 3 4 5
Sum: 15
```

```
[ohyong@cse ~/cp/Lab11]$ ./ex11_2
Enter number of elements: 3
Enter 3 integers: 10 20 30
Sum: 60
```

```
[ohyong@cse ~/cp/Lab11]$ ./ex11_2
Enter number of elements: 4
Enter 4 integers: -1 0 1 2
Sum: 2
```

Ex3

- Take input from the user for the number of rows *r* and the number of columns *c*, *dynamically allocate* a two-dimensional integer array of size $r \times c$, and output the total sum and average by taking input values.

Ex3

- **Program output**

```
[ohyong@cse ~/cp/Lab11]$ vi ex11_3.c
[ohyong@cse ~/cp/Lab11]$ gcc ex11_3.c -o ex11_3
[ohyong@cse ~/cp/Lab11]$ ./ex11_3
Enter number of rows: 2
Enter number of columns: 3
Enter 6 integers:
1 2 3 4 5 6
Sum: 21
Average: 3.50
```

```
[ohyong@cse ~/cp/Lab11]$ ./ex11_3
Enter number of rows: 4
Enter number of columns: 3
Enter 12 integers:
1 2 3
4 5 6
7 8 9
10 11 12
Sum: 78
Average: 6.50
```

Extra

(Inner product)

- Assume two vectors $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$. The *inner product* between **a** and **b** is defined as

$$\mathbf{a} \cdot \mathbf{b} = a_0 b_0 + a_1 b_1 + \ \ldots + a_{n-1} b_{n-1}$$

- For given **a** and **b**, compute the inner product of $\mathbf{a} \cdot \mathbf{b}$.

Extra

- **Program setup**
  1) Ask the user to enter two vectors of length n = 5, and store them in arrays **`a`** and **`b`** (**`float`**).
  2) **Implement a function** that returns the inner product between **`a`** and **`b`**. The array names are passed as *pointers* to the function.
  3) Display the result.

  The prototype of the functions is as follows.

  - float inn_prod(float* a, float* b, int len);

Extra

- **Program output**

```
[ohyong@cse ~/cp/Lab11]$ vi ex11_extra.c
[ohyong@cse ~/cp/Lab11]$ gcc ex11_extra.c -o ex11_extra
[ohyong@cse ~/cp/Lab11]$ ./ex11_extra

Enter a vector a: -1 1 1 -1 -1
Enter a vector b: 1 1 -1 1 -1

The inner product is -1.000
```