8주차 실습

2017-10-26

Topic

- String
- Memory allocation
- Sorting

scanf (문자열 입력 받기)

Input Value

programming practice

Output Value

programming

문제의 조건에서 n 값이 주어지지 않았을 때 (n < 1000) [n : 입력 받을 문자열의 길이]

```
char str1[1000]; /* 이때 n이 주어진다면 malloc 함수를 통해
n개만큼 받을 수 있도록 설정할 수 있습니다. 추후 설명*/
scanf("%s", str);
printf("%s", str);
```

현재까지는 %c를 반복하여 받는 형식으로 문자열을 받았지만, **%s**를 이용하면 이어진 문자열 (space bar, enter key, tab key로 구분된)을 **한번에 받을 수 있습니다**.

malloc(size_t size);

- #include<stdlib.h>를 한 뒤 사용가능
- 포인터에 동적으로 주소를 할당할 수 있게 하는 함수
- 입력받은 크기의 공간만큼 시스템에서 할당하고, 그 공간의 첫 부분의 주소를 반환한다.
- Ex)

```
int* intPtr;
intPtr = (int*) malloc(sizeof(int) * 12); //intPtr에 12개의 int가 들어갈 공간을 할당함.
intPtr[0] = 10; // intPtr[1], intPtr[2], ... ,intPtr[11] 모두 값을 선언할 수 있게 됨.
```

sizeof 함수

- sizeof(type) : type의 데이터 크기를 리턴

- Ex)

sizeof(char) : 1

sizeof(int) : 4

int intArray[10];

sizeof(intArray) : 40

scanf 와 malloc의 사용 예시

Input Value

```
5
10 12 13 15 20
```

```
int n;
int* intPtr;
scanf("%d", &n);
intPtr = (int*) malloc( sizeof(int) * n); //int n개만큼의 크기
for(int i=0; i<n; i++)
{
    scanf("%d", &intPtr[i]);
}
```

intPtr에 정확히 integer n개를 받을 수 있는 공간이 생겼고, 이를 scanf로 값을 입력해주는 코드입니다.

HW 1 : Counting

Description

You are given a sentence that contains capital & small letter and space. The length of each word is shorter than 15 letters.

Each word is separated by one(or more) space(s).

Write a program to count number of words in sentence.

Input

The input contains a string of sentence. The length of entire sentence is less then 10,000

Output

Output a single integer word count.

HW 1 : Counting

Example

Input	Output
A B C D E F G	7
the lazy fox jumped over the quick brown dog	9

HW 2 : String functions

Description

Implement ‘my_strlen()’, and ‘my_strrev()’ function.

‘my_strlen()’ function takes a character pointer variable as a parameter, and returns a length of the parameter.

‘my_strrev()’ function takes two character pointer variables str1 and str2. This function reverses the str1 and store it to the str2.

Caution

You should not use built-in functions in <string.h> like strlen(), strcmp(), strchr().

Function Prototype

int my_strlen(char* str);

void my_strrev(char* str1, char* str2);

HW 2 : String functions

Test Code

```
int my_strlen(char *str){
        // you should implement this function
}
void my_strrev(char *str1, char *str2){
            // you should implement this function
}
// lines below this must be commented out when submitting your code.
int main(){
    char str1[100] = "Arsenal";
    char str2[100];
    my_strrev(str1, str2);
    printf("%d", my_strlen(str1)) // "7"
    printf("%s", str2); // "lanesrA"
}
```

HW 3 : Let’s practice malloc()

Description

Implement a ‘number_malloc()’ function.

‘number_malloc()’ function takes an integer, and returns an integer pointer which points to an array {1,2,…n}

Function Prototype

int* number_malloc(int n)

HW 3 : Let’s practice malloc()

Test Code

```
int* number_malloc(int n){
        // you should implement this function
}

// lines below this must be commented out when submitting your code.
int main(){
        int* arr;
        int num;
        scanf("%d", &num);
        arr = number_malloc(num);
        int i;
        for(i = 0 ; i < num ; i++){
                printf("%d ",arr[i]);
        }
}
```

Input	Output
5	1 2 3 4 5
10	1 2 3 4 5 6 7 8 9 10

HW 4 : Sort the array

Description

Implement a ‘sort_array()’ function.

‘sort_array()’ function takes two parameters, an integer(array’s size) and an integer pointer(array to bes orted), then it returns an integer pointer pointing to array sorted in **decreasing order** .

Int n : array size

Int* arr : array to be sorted

Function Prototype

int* sort_array(int n, int* arr1)

HW 4 : Sort the array

Test Code

```
int* sort_array(int n, int* arr){
        // you should implement this function
}

// lines below this must be commented out when submitting your code.
int main(){
        int* sorted;
        int original[10] ={2,1,3,4,5,8,10,7,6,9};
        sorted = sort_array(10,original);
        int i ;
        for( i = 0 ; i < 10 ; i++){
        printf("%d ",sorted[i]);
        }
}
```

original	sort_array(n,original)
{2,1,3,4,5,8,10,7,6,9}	10 9 8 7 6 5 4 3 2 1
{1,2,3,4,5}	5 4 3 2 1