자료구조

김태원

2023년 10월 3일

제 1 장

C 리뷰

1. 입력으로 하나의 양의 정수를 받은 후 0이 될 때까지 연속적으로 2로 나눈 몫을 출력하는 프로그램을 작성하라.

```c
#include <stdio.h>
int main(){
  int n;
  scanf("%d", &n);
  for(int i=n; i!=0; i/=2)
    printf("%d ", i);
}
```

쉬운 문제다. `i/=2`는 그냥 멋내는 용이다.

2. 입력으로 하나의 양의 정수 n을 받은 후 다음의 합을 구하여 출력하는 프로그램을 작성하라. 단, 소수점 4자리까지만 출력하라.

$$1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$$

```c
#include <stdio.h>
int main(){
  int n;
  double sum = 0;
  for(double r=1; r<=n; r++)
    sum = sum + 1/r;
  printf("%.4f\n", sum);
}
```

5번 라인의 `for`문에서 `r`을 `double`로 선언했다. `int`로 선언하면 모든 입력에 대해 출력은 1이기 때문이다.

3. 입력으로 하나의 양의 정수 n을 받은 후 다음의 합을 구하여 출력하는 프로그램을 작성하라.

단, 소수점 4자리까지만 출력하라.

$$1+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}$$

```c
#include <stdio.h>
int main(){
  int n;
  double sum = 0;
  scanf("%d", &n);
  for(double i=1; i<=n; i++){
    double frac = 1;
    for(double j=i; j>0; j--)
      frac *= j;
    sum += 1/frac;
  }
  printf("%.4f\n", sum);
}
```

순서대로 해결했다. 9번 라인이 아니라 7번 라인에서 `frac`을 선언한 이유는 10번 라인에서 `frac`이 쓰이기 때문이다.

4. 먼저 입력될 정수의 개수 $n \leq 100$을 입력받고, 이어서 n개의 정수를 받아 평균과 표준편차를 계산하여 소수점 이하 4번째 자리까지 출력하는 프로그램을 작성하라. 표준편차는 다음과 같이 정의된다. 루트를 계산하기 위해서 `math.h`를 `include`하고 `sqrt`함수를 사용하라.

$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^2}$$

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(){
  int n;
  int *arr;
  double avg = 0;
  double sum = 0;
  scanf("%d", &n);
  arr = (int *)malloc(sizeof(int)*n);
  for(int i=0; i<n; i++){
    int p;
    scanf("%d", &p);
```

```
14      arr[i] = p;
15      avg += arr[i];
16    }
17    for(int i=0; i<n; i++)
18      sum += pow(arr[i]-avg/n, 2.0);
19    printf("%.5g\n", sqrt(sum/n));
20    free(arr);
21  }
```

.5g라는 formant specifier를 사용했는데, 가령 4.5000...를 4.5로 잘라서 출력하기 위해서다. 참고로 `math.h` 라이브러리를 포함한 파일을 컴파일하려면 `-lm` 명령어를 덧붙여야 한다.

5. 먼저 입력될 정수의 개수 $2 \leq n \leq 100$을 입력받고, 이어서 n개의 정수를 입력받는다. 입력된 정수들 중에서 최소값과 두 번째로 작은 값을 찾아 출력하는 프로그램을 작성하라. 만약 최소값이 2개 이상 중복되어 존재하면 그 중 하나를 최소값으로, 다른 하나를 두 번째로 작은 값으로 간주한다.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main(){
4    int n, key;
5    int *arr;
6    scanf("%d", &n);
7    arr = (int*)malloc(sizeof(int)*n);
8    for(int i=0; i<n; i++)
9      scanf("%d", &arr[i]);
10   for(int i=1; i<n; i++){
11     key = arr[i];
12     int j;
13     for(j=i-1; j>=0 && arr[j]>key; j--)
14       arr[j+1] = arr[j];
15     arr[j+1] = key;
16   }
17   printf("%d %d\n", arr[0], arr[1]);
18   free(arr);
19 }
```

10번 라인에서 시작되는 `for` 블록은 삽입정렬 알고리즘을 구현한 것이다. 삽입 정렬 알고리즘은 아

Algorithm 1: Insertion Sort

Input: A: Array
for $i \in \{2, \ldots, A.length\}$ **do**
  $key \leftarrow A[i]$
  $j \leftarrow i - 1$
  while $j > 0$ & $A[j] > key$ **do**
    $A[j+1] \leftarrow A[j]$
    $j \leftarrow j - 1$
  $A[j+1] \leftarrow key$

래와 같은 방식으로 작동한다.

$A = [7, 3, 1, 2, 4, 6]$ 첫 번째 for 루프: $7 = A[1] > key = A[2] = 3$

$\mapsto [3, 7, 1, 2, 4, 6]$

두 번째 for 루프: $7 = A[2] > key = A[3] = 1$

$\mapsto [3, 1, 7, 2, 4, 6]$

두 번째 for 루프: $3 = A[1] > key = A[3] = 1$

$\mapsto [1, 3, 7, 2, 4, 6]$

세 번째 for 루프: $7 = A[3] > key = A[4] = 2$

$\mapsto [1, 3, 2, 7, 4, 6]$

세 번째 for 루프: $3 = A[2] > key = A[4] = 2$

$\mapsto [1, 2, 3, 7, 4, 6]$

네 번째 for 루프: $7 = A[4] > key = A[5] = 4$

$\mapsto [1, 2, 3, 4, 7, 6]$

다섯 번째 for 루프: $7 = A[5] > key = A[6] = 6$

$\mapsto [1, 2, 3, 4, 6, 7]$

6. 수열에서 큰 값이 작은 값보다 앞서 나오는 경우 두 값을 역전된 쌍이라고 부른다. 예를 들어 수열 $4, 2, 1, 1, 3$에는 $(4, 2), (4, 1), (4, 1), (4, 3), (2, 1), (2, 1)$의 총 6개의 역전된 쌍이 있다. 수열을 입력으로 받아서 역전된 쌍의 개수를 카운트하여 출력하는 프로그램을 작성하라. 키보드로부터 먼저 정수의 개수 N을 입력받고, 이어서 N개의 정수를 입력 받는다.

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  int n, gncount;
  int* arr;
  scanf("%d", &n);
  arr = (int*)malloc(sizeof(int)*n);
```

```
  for(int i=0; i<n; i++)
    scanf("%d", &arr[i]);
  for(int i=0; i<n; i++){
    int gn = arr[i];
    for(int j=i; j<n; j++){
      if(gn > arr[j])
        gncount++;
    }
  }
  free(arr);
  printf("%d\n", gncount);
}
```

for문을 돌면서 각 성분에 대해 그 성분보다 큰 것들을 세는 단순무식한 코드다.

7. 키보드로부터 연속해서 음이 아닌 정수들을 입력받는다. 정수가 하나씩 입력될 때마다 현재까지 입력된 정수들을 오름차순으로 정렬하여 화면에 출력한다. 단, 새로 입력된 정수가 이미 이전에 입력된 정수라면 duplicate라고 출력하고 저장하지 않고 버린다. 사용자가 -1을 입력하면 프로그램을 종료한다. 입력되는 정수의 개수는 100개를 넘지 않는다.

```
#include <stdio.h>
int check(int* arr, int size, int n);
void insertionSort(int* arr, int size);
int main(){
  int arr[100];
  scanf("%d", &arr[0]);
  printf("%d\n\n", arr[0]);
  for(int i=1; i<100; i++){
    int n;
    scanf("%d", &n);
    if(n==-1)
      break;
    if(check(arr, i, n)){
      arr[i] = n;
      insertionSort(arr, i);
      for(int j=0; j<=i; j++)
        printf("%d ", arr[j]);
    }
    else{
      i--;
      printf("duplicate");
    }
```

```
    printf("\n\n");
  }
}
int check(int* arr, int size, int n){
  int result;
  for(int i=0; i<size; i++){
    if(arr[i]==n){
      result = 0;
      break;
    }
    else
      result = 1;
  }
  return result;
}
void insertionSort(int* arr, int size){
  for(int i=1; i<=size; i++){
    int key = arr[i];
    int j;
    for(j=i-1; j>=0 && arr[j]>key; j--)
      arr[j+1] = arr[j];
    arr[j+1] = key
  }
}
```

8. 먼저 입력될 정수의 개수 $n \leq 100$을 입력받고, 이어서 n개의 정수를 받아 순서대로 배열에 저장한다. 그런 다음 키보드로부터 다시 하나의 정수 k를 입력받은 후 n개의 정수들 중에서 k에 가장 가까운, 즉 k와의 차이의 절대값이 가장 작은 정수를 찾아 출력하는 프로그램을 작성하라.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  int n, k, diff;
  int* arr;
  scanf("%d", &n);
  arr = (int*)malloc(sizeof(int)*n);
  for(int i=0; i<n; i++)
    scanf("%d", &arr[i]);
  scanf("%d", &k);
  diff = arr[0];
```

```
  for(int i=0; i<n; i++){
    if(abs(k-diff) > abs(k-arr[i]))
      diff = arr[i];
  }
  free(arr);
  printf("%d\n", diff);
}
```

abs 함수를 사용한 단순무식한 방법이다.

9. 사용자로부터 $n < 100$개의 정수를 입력받아 크기순으로 정렬한 후 중복된 수를 제거하는 프로그램을 작성하라. 입력 형식은 먼저 n의 값이 주어지고 이어서 n개의 정수들이 주어진다. 예를 들어 $n = 8$이고 입력된 정수들이 $4, 7, 4, 12, 410, 9, 7$이라면 중복을 제거하고 남은 정수들은 $4, 7, 9, 10, 12$이다. 그러면 먼저 남은 정수의 개수 5를 출력하고 콜론을 출력한 후 남은 정수들을 오름차순으로 출력한다.

```
#include <stdio.h>
#include <stdlib.h>
void insertionSort(int* arr, int size);
int main(){
  int n;
  int count = 0;
  int index = 0;
  int *temp, *arr;
  scanf("%d", &n);
  temp = (int*)malloc(sizeof(int)*n);
  arr = (int*)malloc(sizeof(int)*count);
  for(int i=0; i<n; i++)
    scanf("%d", &temp[i]);
  for(int i=0; i<n; i++){
    int check = 0;
    for(int j=0; j<i; j++){
      if(temp[i]==temp[j]){
        check = 1;
        break;
      }
    }
    if(check == 0){
      count++;
      arr[index] = temp[i];
      index++;
    }
```

```
  }
  free(temp);
  insertionSort(arr, count);
  printf("%d: ", count);
  for(int i=0; i<count; i++)
    printf("%d ", arr[i]);
  printf("\n");
  free(arr);
}
void insertionSort(int* arr, int size){
  for(int i=1; i<size; i++){
    int key = arr[i];
    int j;
    for(j=i-1; j>=0 && arr[j]>key; j--)
      arr[j+1] = arr[j];
    arr[j+1] = key;
  }
}
```

다양한 불변항을 사용했다.

10. 정렬을 하는 가장 간단한 방법 중의 하나는 다음과 같이 하는 것이다. 배열 data에 data[0]에서 data[n-1]까지 n개의 정수가 저장되어 있다. 먼저 data[0]와 data[n-1] 사이의 정수들 중에서 가장 큰 정수를 찾는다. 그것을 data[k]라고 가정해보자. 그러면 data[k]와 data[n-1]을 교환한다. 이제 가장 큰 정수가 data[n-1], 즉 맨 마지막 위치에 저장되었으므로 그 값에 대해서는 더 이상 생각할 필요가 없다. 이제 data[0] - data[n-2] 중에서 최대값을 찾는다. 그 값을 data[p]라고 하자. 그러면 다시 data[p]와 data[n-2]를 교환하고 data[n-2]에 대해서는 잊어버려도 된다. 이런 식으로 계속하면 마지막에는 data[0]과 data[1] 중에 최대값을 data[1]과 교환하면 전체의 정렬이 완료된다. 이 알고리즘을 구현하라. 입력은 먼저 정렬할 개수 $n \leq 100$이 주어지고 이어서 n개의 정수들이 주어진다.

```
#include <stdio.h>
#include <stdlib.h>
void selectionSort(int* arr, int size);
void swap(int* a, int* b);
int main(){
  int n;
  int* data;
  scanf("%d", &n);
  data = (int*)malloc(sizeof(int)*n);
  for(int i=0; i<n; i++)
    scanf("%d", &data[i]);
```

```
  selectionSort(data, n);
  for(int i=0; i<n; i++)
    printf("%d ", data[i]);
  printf("\n");
  free(data);
}
void selectionSort(int* arr, int size){
  int i, j, max;
  for(i=size-1; i>0; i--){
    max = i;
    for(j=i-1; j>=0; j--)
      if(arr[j] > arr[max])
        max=j;
      swap(&arr[max], &arr[i]);
  }
}
void swap(int* a, int* b){
  int temp = *a;
  *a = *b;
  *b = temp;
}
```

선택정렬 알고리즘에 대한 문제다. 배열 성분 자체를 최대값으로 두지 않고 최대값의 성분에 대응하는 인덱스로 정렬한다는 점에 유의하라. 보통 선택 정렬은 최대값이 아니라 최소값을 기준으로 한다. 불변항만 살짝 바꾸면 된다.

Algorithm 2: Selection Sort

Input: A: Array
for $i \in \{1, \ldots, A.length - 1\}$ **do**
  $min \leftarrow i$
  for $j \in \{i+1, A.length - 1$ **do**
    if $A[j] < A[min]$ **then**
      $min \leftarrow j$
  if $min \neq i$ **then**
    swap$(A[i], A[min])$

11. 입력으로 하나의 문자열을 받은 후 뒤집어서 출력하는 프로그램을 작성하라. 예를 들어 `hello`를 입력하면 `olleh`가 출력된다.

```
#include <stdio.h>
#include <string.h>
void revstr(char *str);
int main(){
```

```
  char* str;
  scanf("%s", str);
  revstr(str);
  printf("%s\n", str);
  return 0;
}
void revstr(char *str){
  int len = strlen(str);
  for(int i=0; i<len/2; i++){
    int temp = str[i];
    str[i] = str[len-i-1];
    str[len-i-1] = temp;
  }
}
```

단순하게 swap하여 해결했다.

12. 영문 소문자로 구성된 하나의 문자열을 입력받은 후 문자열을 구성하는 문자들을 알파벳 순으로 정렬하여 만들어지는 문자열을 출력하라. 예를 들어 hello가 입력되면 ehllo를 출력한다.

```
#include <stdio.h>
#include <string.h>
void alphabetOrder();
int main(){
  char *ch;
  scanf("%s", ch);
  alphabetOrder(ch);
  puts(ch);
}
void alphabetOrder(char *ch){
  char temp;
  int i, j, length = strlen(ch);
  for(i=0; i<length; i++){
    for(j=i+1; j<length; j++){
      if(ch[i] > ch[j]){
        temp = ch[i];
        ch[i] = ch[j];
        ch[j] = temp;
      }
    }
  }
```

```
}
```

기본적으로 선택정렬 알고리즘의 응용이다.

13. 아나그램이란 문자들의 순서를 재배열하여 동일하게 만들 수 있는 문자열을 말한다. 대소문자는 구분하지 않는다. 예를 들어서 Silent와 Listen은 아나그램이다. 입력으로 두 문자열을 받아서 아나그램인지 판단하는 프로그램을 작성하라.

```c
#include <stdio.h>
#include <string.h>
void cnvt_lwr(char *str);
int anagram(char *str1, char *str2);
int main(){
  char *words[2];
  for(int i=0; i<2; i++){
    char buf[100];
    scanf("%s", buf);
    words[i] = strdup(buf);
    cnvt_lwr(words[i]);
  }
  if(anagram(words[0], words[1]))
    puts("yes");
  else
    puts("no");
}
void cnvt_lwr(char *str){
  for(int i=0; i<strlen(str); i++){
    if(str[i] >= 65 && str[i] <= 90)
      str[i] = str[i]+32;
  }
}
int anagram(char *str1, char *str2){
  if(strlen(str1)!=strlen(str2))
    return 0;
  int count = 0;
  for(int i=0; i<strlen(str1); i++){
    for(int j=0; j<strlen(str1); j++){
      if(str1[i]==str2[j]){
        count++;
        break;
      }
    }
```

```
  }
  if(count==check)
    return 1;
  else
    return 0;
}
```

main에서 중요한 것은 buf와 strdup을 이용한 입력 이다. anagram은 단순무식한 논리를 break로 구현한 것이고 cnvt_lwr는 ASCII 값을 활용하는 함수이므로 그냥 외워두는 편이 나을 것 같다.

14. 영문 소문자로 구성된 2개의 단어를 입력받은 후 두 단어가 동일한 문자집합으로 구성되었는지 검사하여 yes 혹은 no를 출력하는 프로그램을 작성하라. 예를 들어 ababc와 cba는 문자집합 {a,b,c}로 구성되었으므로 yes다. 입력 단어의 길이는 20이하다.

```
#include <stdio.h>
#include <string.h>
int alphSet(int* wordA, int* wordB);
int main(){
  char* words[2];
  char alphabet[26];
  int firstAlphCount[26], secAlphCount[26];
  for(int i=0; i<2; i++){
    char buf[20];
    scanf("%s", buf);
    words[i] = strdup(buf);
  }
  for(int i=0; i<26; i++){
    alphabet[i] = 'a'+i;
    firstAlphCount[i] = 0;
    secAlphCount[i] = 0;
  }
  for(int i=0;i<strlen(words[0]);i++){
    for(int j=0;j<26;j++){
      if(words[0][i] == alphabet[j])
        firstAlphCount[i] = 1;
    }
  }
  for(int i=0;i<strlen(words[1]);i++){
    for(int j=0;j<26;j++){
      if(words[1][i] == alphabet[j])
        secAlphCount[i] = 1;
    }
  }
```

```
30    if(alphSet(firstAlphCount, secAlphCount))
31      puts("yes");
32    else
33      puts("no");
34  }
35  int alphSet(int* wordA, int* wordB){
36    for(int i=0; i<26; i++){
37      if(wordsA[i] != wordsB[i])
38        return 0;
39    }
40    return 1;
41  }
```

단순무식하지만 뭘 하려는지 잘 보인다. 아무튼 14번 라인인 `alphabet[i] = 'a'+i`가 핵심이다.

15. 입력으로 $n < 100$개의 영문 문자열을 받는다. 각 문자열의 길이는 20이하다. 이 문자열들을 문자열의 길이가 짧은 것부터 긴 것 순서로 정렬하여 출력하라. 단, 길이가 동일한 문자열들은 그들끼리 사전식 순서로 정렬해야 한다. 입력 형식은 먼저 문자열의 개수 n이 주어지고, 이어서 n개의 문자열이 한 줄에 하나씩 주어진다.

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   int main(){
5     int n;
6     char **words;
7     scanf("%d", &n);
8     words = (char **)malloc(sizeof(char[101])*n);
9     for(int i=0; i<n; i++){
10      char buf[101];
11      scanf("%s", buf);
12      words[i] = strdup(buf);
13    }
14    for(int i=0; i<n; i++){
15      for(int j=0; j<n; j++){
16        if(strlen(words[j])>strlen(words[i])){
17          char* temp1;
18          temp1 = words[i];
19          words[i] = words[j];
20          words[j] = temp1;
21        }
22        else if(strlen(words[j])==strlen(words[i])){
```

```
23          if(words[j][0] > words[i][0]){
24            char* temp2;
25            temp2 = words[i];
26            words[i] = words[j];
27            words[j] = temp2;
28          }
29        }
30      }
31    }
32    printf("\n");
33    for(int i=0; i<n; i++)
34      printf("%s\n", words[i]);
35    free(words);
36  }
```

8번 라인의 동적 할당에서 쓰인 이중 포인터를 이해하는 것이 관건이다. 다시 말해 14번 라인 이하의 `for` 블록에서 swap되는 `words[]`는 문자가 아니라 문자열 자체다.