

자료구조

김태원

2023년 10월 2일

제1장

C 리뷰

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

```
1 #include <stdio.h>
2 int main(){
3     int n;
4     scanf("%d", &n);
5     for(int i=n; i!=0; i/=2)
6         printf("%d ", i);
7 }
```

쉬운 문제다. `i/=2`는 그냥 멧내는 용이다.

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

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

```
1 #include <stdio.h>
2 int main(){
3     int n;
4     double sum = 0;
5     for(double r=1; r<=n; r++)
6         sum = sum + 1/r;
7     printf("%.4f\n", sum);
8 }
```

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

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

단, 소수점 4자리까지만 출력하라.

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

```
1 #include <stdio.h>
2 int main(){
3     int n;
4     double sum = 0;
5     scanf("%d", &n);
6     for(double i=1; i<=n; i++){
7         double frac = 1;
8         for(double j=i; j>0; j--){
9             frac *= j;
10            sum += 1/frac;
11        }
12        printf("%.4f\n", sum);
13    }
```

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 n 개의 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.

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 int main(){
5     int n;
6     int *arr;
7     double avg = 0;
8     double sum = 0;
9     scanf("%d", &n);
10    arr = (int *)malloc(sizeof(int)*n);
11    for(int i=0; i<n; i++){
12        int p;
13        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 }

```

`.5g` 라는 format 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 n 개의 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

```

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

```

$A = [7, 3, 1, 2, 4, 6]$ 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: $7 = A[5] > key = A[6] = 6$
 $\mapsto [1, 2, 3, 4, 6, 7]$

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     int n, gncount;
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=0; i<n; i++){
11      int gn = arr[i];
12      for(int j=i; j<n; j++){
13          if(gn > arr[j])
14              gncount++;
15      }
16  }
17  free(arr);
18  printf("%d\n", gncount);
19 }

```

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 `duplicate`라고 출력하고 저장하지 않고 버린다. 사용자가 `-1`을 입력하면 프로그램을 종료한다. 입력되는 정수의 개수는 100개를 넘지 않는다.

```

1  #include <stdio.h>
2  int check(int* arr, int size, int n);
3  void insertionSort(int* arr, int size);
4  int main(){
5      int arr[100];
6      scanf("%d", &arr[0]);
7      printf("%d\n\n", arr[0]);
8      for(int i=1; i<100; i++){
9          int n;
10         scanf("%d", &n);
11         if(n==-1)
12             break;
13         if(check(arr, i, n)){
14             arr[i] = n;
15             insertionSort(arr, i);
16             for(int j=0; j<=i; j++)
17                 printf("%d ", arr[j]);
18         }
19         else{
20             i--;
21             printf("duplicate");
22         }

```

```

23     printf("\n\n");
24 }
25 }
26 int check(int* arr, int size, int n){
27     int result;
28     for(int i=0; i<size; i++){
29         if(arr[i]==n){
30             result = 0;
31             break;
32         }
33         else
34             result = 1;
35     }
36     return result;
37 }
38 void insertionSort(int* arr, int size){
39     for(int i=1; i<=size; i++){
40         int key = arr[i];
41         int j;
42         for(j=i-1; j>=0 && arr[j]>key; j--){
43             arr[j+1] = arr[j];
44             arr[j+1] = key;
45         }
46     }

```

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

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 int main(){
4     int n, k, diff;
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    }
11    scanf("%d", &k);
12    diff = arr[0];

```

```

12     for(int i=0; i<n; i++){
13         if(abs(k-diff) > abs(k-arr[i]))
14             diff = arr[i];
15     }
16     free(arr);
17     printf("%d\n", diff);
18 }

```

abs 함수를 사용한 단순무식한 방법이다.

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

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  void insertionSort(int* arr, int size);
4  int main(){
5      int n;
6      int count = 0;
7      int index = 0;
8      int *temp, *arr;
9      scanf("%d", &n);
10     temp = (int*)malloc(sizeof(int)*n);
11     arr = (int*)malloc(sizeof(int)*count);
12     for(int i=0; i<n; i++){
13         scanf("%d", &temp[i]);
14         for(int i=0; i<n; i++){
15             int check = 0;
16             for(int j=0; j<i; j++){
17                 if(temp[i]==temp[j]){
18                     check = 1;
19                     break;
20                 }
21             }
22             if(check == 0){
23                 count++;
24                 arr[index] = temp[i];
25                 index++;
26             }

```



```

27     }
28     free(temp);
29     insertionSort(arr, count);
30     printf("%d: ", count);
31     for(int i=0; i<count; i++)
32         printf("%d ", arr[i]);
33     printf("\n");
34     free(arr);
35 }
36 void insertionSort(int* arr, int size){
37     for(int i=1; i<size; i++){
38         int key = arr[i];
39         int j;
40         for(j=i-1; j>=0 && arr[j]>key; j--)
41             arr[j+1] = arr[j];
42         arr[j+1] = key;
43     }
44 }

```

다양한 불변항을 사용했다.

10. 정렬을 하는 가장 간단한 방법 중의 하나는 다음과 같이 하는 것이다. 배열 `data`에 `data[0]`에서 `data[n-1]`까지 n 개의 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 n 개의 정수들이 주어진다.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 void selectionSort(int* arr, int size);
4 void swap(int* a, int* b);
5 int main(){
6     int n;
7     int* data;
8     scanf("%d", &n);
9     data = (int*)malloc(sizeof(int)*n);
10    for(int i=0; i<n; i++)
11        scanf("%d", &data[i]);

```

```

12  selectionSort(data, n);
13  for(int i=0; i<n; i++)
14      printf("%d ", data[i]);
15  printf("\n");
16  free(data);
17  }
18  void selectionSort(int* arr, int size){
19      int i, j, max;
20      for(i=size-1; i>0; i--){
21          max = i;
22          for(j=i-1; j>=0; j--){
23              if(arr[j] > arr[max])
24                  max=j;
25              swap(&arr[max], &arr[i]);
26          }
27      }
28  void swap(int* a, int* b){
29      int temp = *a;
30      *a = *b;
31      *b = temp;
32  }

```

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 $i \in \{1, \dots, A.length - 1\}$ **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($A[i], A[min]$)
