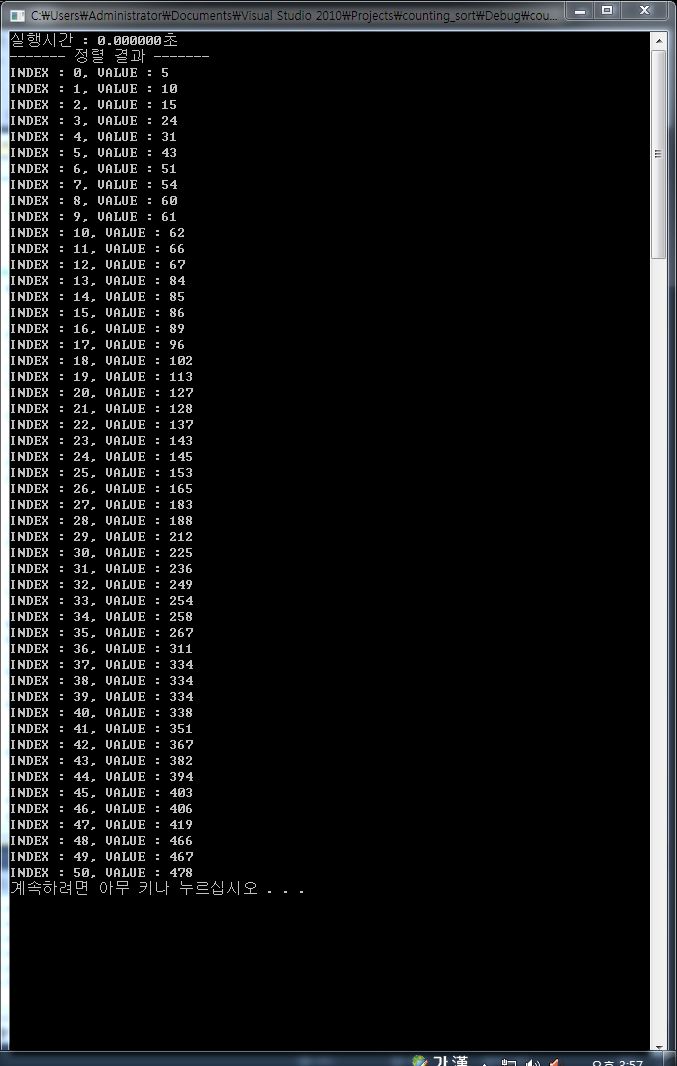
알고리즘 과제

Practice.04

**학번 : 201402432**

**이름 : 조디모데**

**Counting Sort.**

* Input\_Small   
  - 출력 결과  
  
* 알고리즘 설명  
   1. 배열의 가장 큰 값을 크기로 갖는 배열을 만든다.  
   2. 배열의 index와 같은 값을 가진 수를 세어 해당 index에 저장한다.  
   3. 배열의 index를 증가시키며 배열(index) += 배열(index-1) 이런식으로 누적하여 더한 값을 배열에 저장한다.  
   4. 누적된 값의 차이를 이용하여 정렬된 배열을 구한다
* 컴파일 방법  
  input 폴더를 다음 위치에 넣는다  
  C:\\Users\\Administrator\\Desktop\\input\\input50.txt
* 수행시간을 그래프

Input  
50 : 0  
500 : 0.00  
5000 : 0.0005  
10000 : 0.001

* Code ( .C )

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <memory.h>

int getMaxValue(int arr[],int length)

{

int maxValue, i ;

maxValue = 0 ;

for(i=0 ; i<length ; i++){

if(arr[i] > maxValue)

maxValue = arr[i] ;

}

return maxValue ;

}

int\* countingSort(int maxValue, int arr[],int length)

{

int i, j, value, arrIndex, count ;

int\* bucket ;

bucket = (int\*)malloc(sizeof(int)\*(maxValue+1)) ;

memset(bucket, 0, sizeof(int)\*maxValue+1) ;

for(i=0 ; i<=length ; i++){

bucket[ arr[i] ]++ ;

}

for(i=1 ; i<maxValue ; i++){

bucket[i] += bucket[i-1] ;

}

arrIndex = 0 ;

for(i=1 ; i<maxValue ; i++){

if(bucket[i]!=bucket[i-1]){

for(j=bucket[i] ; j>bucket[i-1] ; j--){

arr[arrIndex] = i ;

arrIndex ++ ;

}

}

}

return arr ;

}

void printArray(int\* arr,int length)

{

int i ;

for(i=0 ; i<length ; i++){

//if(arr[i]!=0 && arr[i]>=0)

printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;

}

}

// main function

int main(void)

{

int i,n, maxValue, num[51] ;

FILE \*fps ;

int temp = 0 ;

int \*bucket ;

clock\_t start,end ;

n = 51 ;

// input의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input50.txt","rt");

for(i=0; i <= n ; i++){

fscanf(fps,"%d",&temp) ;

num[i]=temp ;

}

fclose(fps) ;

// 정¢´렬¤A 시öA작U 시öA간Æ¡Ì 저u장a

start = clock() ;

maxValue = getMaxValue(num, sizeof(num)/sizeof(int)) ;

bucket = (int\*)malloc(sizeof(int)\*(maxValue+1)) ;

// 정¢´렬¤A하I는¥A 부¬I분¬¨¢

bucket = countingSort(maxValue, num, n) ;

// 정¢´렬¤A 후A 시öA간Æ¡Ì 저u장a

end = clock() ;

// 실öC행a 시öA간Æ¡Ì 출a력¤A

printf("실öC행a시öA간Æ¡Ì : %lf초E\n",(end-start)/(double)1000) ;

// 정¢´렬¤A 결Æa과Æu 출a력¤A

printf("------- 정¢´렬¤A 결Æa과Æu ------- \n") ;

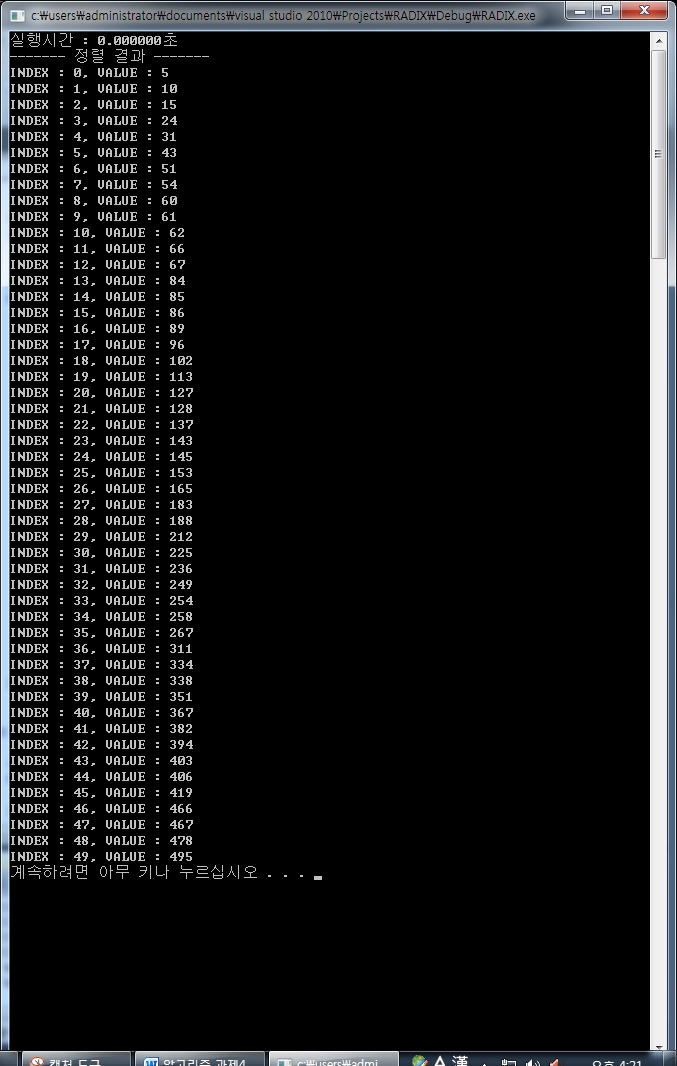
printArray(bucket, n) ;

system("pause") ;

return 0;

}

**Radix Sort.**

* Input50   
  - 출력 결과  
  
* 알고리즘 설명  
   숫자의 특정 자릿수만을 보고 정렬하는 방법이다.  
  최대 자리수가 3이라고 하면 우선 1의 자리, 10의 자리, 100의 자리 순으로 오름차순 정렬을 한다.
* 컴파일 방법  
  input 폴더를 바탕화면에 넣는다  
  C:\\Users\\Administrator\\Desktop\\input\\input50.txt  
    
  원하는 데이터의 크기에 맞게 num[크기+1] 배열 초기화  
  n을 원하는 크기에 맞게 설정
* 수행시간을 그래프

Input  
50 : 0.0000  
500 : 0.0002  
5000 : 0.0008  
10000 : 0.0018

* Code ( .C )

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <memory.h>

#include <math.h>

void radixSort(int \*data, int size, int p, int k) {

int \*count, \*tmp ;

int index, pval, i, j, n;

if ( (count=(int\*)malloc(k\*sizeof(int))) == NULL ){

memset(count, 0, sizeof(int)\*k) ;

return;

}

if ( (tmp=(int\*)malloc(size\*sizeof(int))) == NULL ){

memset(tmp, 0, sizeof(int)\*size) ;

return;

}

for (n=0; n<p; n++) {

for (i=0; i<k; i++)

count[i] = 0;

// 위¡×치¢® 계Æe산íe

pval = (int)pow((double)k, (double)n);

for (j=0; j<size; j++) {

index = (int)(data[j] / pval) % k;

count[index] = count[index] + 1;

}

for (i=1; i<k; i++) {

count[i] = count[i] + count[i-1];

}

// 계Æe수ùo정¢´렬¤A 방ö©¡식öA

for (j=size-1; j>=0; j--) {

index = (int)(data[j] / pval) % k;

tmp[count[index] -1] = data[j];

count[index] = count[index] - 1;

}

memcpy(data, tmp, size \* sizeof(int));

}

free(count) ;

free(tmp) ;

}

int count\_large(int arr[], int n){

int i, j, big ;

big = arr[0] ;

for(i=1 ; i<n ; i++){

if(big < arr[i])

big = arr[i] ;

}

j=0 ;

while(big != 0){

big= big/10 ;

j++ ;

}

return j ;

}

void printArray(int\* arr,int length)

{

int i ;

for(i=0 ; i<length ; i++){

printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;

}

}

int main(void)

{

int i,n, maxValue, num[500] ;

FILE \*fps ;

int temp = 0 ;

clock\_t start,end ;

n = 500 ;

// input의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input500.txt","rt");

for(i=0; i <= n ; i++){

fscanf(fps,"%d",&temp) ;

num[i]=temp ;

}

fclose(fps) ;

temp = count\_large(num, n) ;

// 정¢´렬¤A 시öA작U 시öA간Æ¡Ì 저u장a

start = clock() ;

// 정¢´렬¤A하I는¥A 부¬I분¬¨¢

radixSort(num, n, temp, 10) ; // (배öe열¯¡©, 배öe열¯¡©크¨Ï기¾a, 최O대¥e 자U리¬¢ç수ùo, 진©ª수ùo)

// 정¢´렬¤A 후A 시öA간Æ¡Ì 저u장a

end = clock() ;

// 실öC행a 시öA간Æ¡Ì 출a력¤A

printf("실öC행a시öA간Æ¡Ì : %lf초E\n",(end-start)/(double)1000) ;

// 정¢´렬¤A 결Æa과Æu 출a력¤A

printf("------- 정¢´렬¤A 결Æa과Æu ------- \n") ;

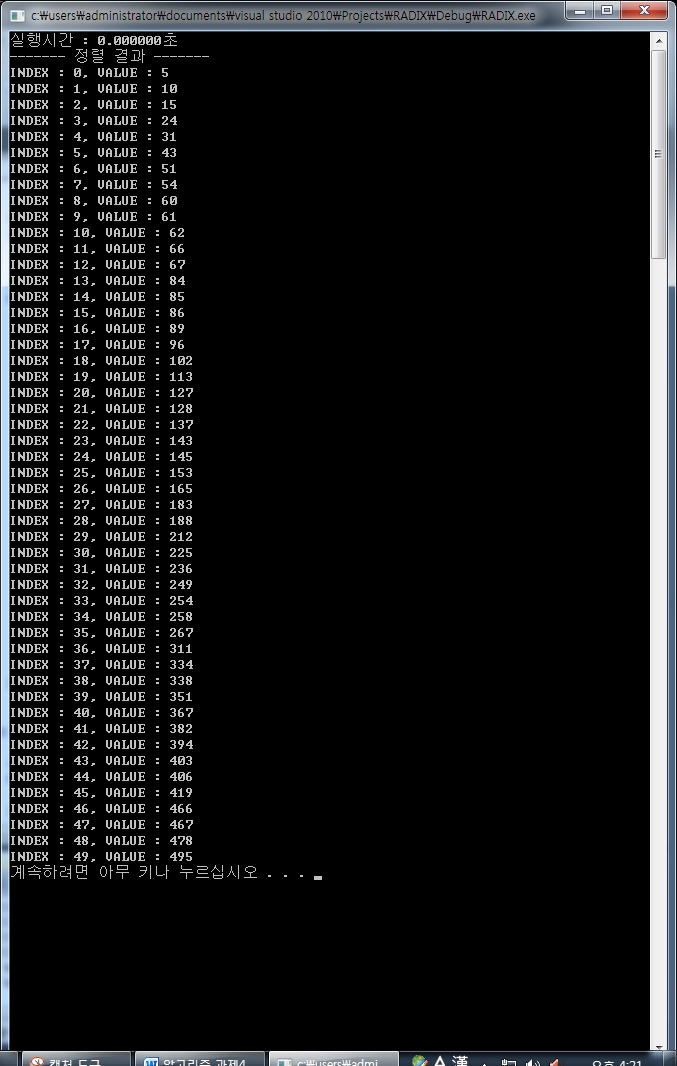
//printArray(num, n) ;

system("pause") ;

return 0;

}

**Bucket sort.**

* Input50   
  - 출력 결과  
  
* 알고리즘 설명  
  배열의 원소들을 수의 범위 별로 구분해 버킷에 넣고 그 버킷 각각을 정렬하는 방법
* 컴파일 방법  
  input 폴더를 바탕화면에 넣는다  
  C:\\Users\\Administrator\\Desktop\\input\\input50.txt  
    
  원하는 데이터의 크기에 맞게 num[크기+1] 배열 초기화  
  n을 원하는 크기에 맞게 설정
* 수행시간을 그래프

Input  
50 : 0.0000  
500 : 0.0012  
5000 : 0.0046  
10000 : 0.0062

* Code ( .C )

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <memory.h>

// 리¬¢ç스ö¨¬트¡¢ç 노øe드ìa

typedef struct node {

int value;

struct node \*link;

} node;

int\* buketSort(int \*ar, int size, int max) {

node \*counter, \*n2,\*n1;

int \*fa, temp ;

int i,j,k=0 ;

int n,a;

fa = (int\*)malloc(sizeof(int)\*(size)) ;

memset(fa, 0, sizeof(int)\*size) ;

max = max\*10+1 ; // count배öe열¯¡©의C 크¨Ï기¾a ;

counter = (node\*)malloc(sizeof(node)\*(max)) ;

for(i=0;i<max ;i++) { // init

counter[i].value = 0;

counter[i].link = 0;

}

for(i=0;i<size ;i++) {

// init

n = ar[i];

j = n \* 100;

j = j/10;

// 버öo킷¢Ò에¯¢® 원¯©ª소ùO가Æ¢® 없ú©ª을¡í때Ò¡×

if(counter[j] . value ==0 && counter[j] . link == 0)

counter[j] . value = ar[i] ;

else {

// 버öo킷¢Ò의C 원¯©ª소ùO가Æ¢® 하I나ø¨£일I 경Æ©¡우¯i

if(counter[j].link==0 && counter[j] .value != 0) {

counter[j].link=(node \*) malloc(sizeof(node));

n2 = counter[j].link;

n2 -> link = 0;

n2 -> value =ar[i];

continue;

}

n2 = counter[j].link ;

while(n2 -> link !=0 ) { // 맨¬C 마¬¢Ò지o막¬¡¤ 노øe드ìa

n2 = n2 -> link;

}

n2 -> link =(node \*) malloc(sizeof(node));

n2 = n2 -> link;

n2 -> link=0;

n2 -> value = ar[i];

}

}

// 버öo킷¢Ò을¡í 순ù©ª서ù¡©대¥e로¤I 정¢´렬¤A

for(i=0;i<max ;i++) {

// 버öo킷¢Ò에¯¢® 노øe드ìa가Æ¢® 없ú©ª을¡í경Æ©¡우¯i

if(counter[i] . link ==0 && counter[i] . value == 0)

continue;

else {

n1 = &counter[i];

n2 = &counter[i] ;

// 버öo킷¢Ò의C 원¯©ª소ùO 개Æ©ø수ùo가Æ¢® 2이I상ío

if(n2 -> link != 0) {

// 버öo킷¢Ò별¬¡Æ 정¢´렬¤A

while(n1!=0) {

while(n2!= 0) {

if(n1 -> value > n2 -> value) {

temp =n1 -> value;

n1 -> value =n2 -> value;

n2 -> value =temp;

}

n2 = n2 -> link;

}

n2 = n1 -> link;

n1 = n1 -> link;

}

n1 = &counter[i];

// 차¡À례¤E대¥e로¤I 담¥a기¾a

for(; n1!=0; k++) {

fa[k] = n1 -> value;

n1 = n1 -> link;

}

}

// 버öo킷¢Ò에¯¢® 하I나ø¨£의C 노øe드ìa가Æ¢® 있O는¥A 경Æ©¡우¯i

else {

fa[k] = counter[i].value;

k=k+1;

}

}

}

return fa;

}

int getMaxValue(int \*arr, int size){

int max = 0, i ;

for(i=0 ; i<size ; i++){

if(arr[i]>max)

max = arr[i] ;

}

return max ;

}

void printArray(int\* arr,int length)

{

int i ;

for(i=0 ; i<length ; i++){

printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;

}

}

int main(void) {

int i,size , maxValue, num[51] ;

FILE \*fps ;

int temp = 0 ;

int \*result ;

clock\_t start,end ;

size = 50 ;

// input의C 숫ùy자U들ìe을¡í 배öe열¯¡© num에¯¢® 저u장a하I는¥A 부¬I분¬¨¢

fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input50.txt","rt");

for(i=0; i <= size ; i++){

fscanf(fps,"%d",&temp) ;

num[i]=temp ;

}

fclose(fps) ;

// 정¢´렬¤A 시öA작U 시öA간Æ¡Ì 저u장a

start = clock() ;

temp = getMaxValue(num, size) ;

// 정¢´렬¤A하I는¥A 부¬I분¬¨¢

result = buketSort(num, size, temp) ;

// 정¢´렬¤A 후A 시öA간Æ¡Ì 저u장a

end = clock() ;

// 실öC행a 시öA간Æ¡Ì 출a력¤A

printf("실öC행a시öA간Æ¡Ì : %lf초E\n",(end-start)/(double)1000) ;

// 정¢´렬¤A 결Æa과Æu 출a력¤A

printf("------- 정¢´렬¤A 결Æa과Æu ------- \n") ;

printArray(result, size) ;

free(result);

system("pause") ;

return 0;

}