Chapter 8. Function

Exercise L8-1 (반올림)

1 #include <stdio.h>  
 2 int round2int(double x);  
 3 int main(void){  
 4 int res;  
 5 double x;  
 6 printf("Put a real number x(-10.0<=x<=10.0) : ");  
 7 scanf("%lf", &x);  
 8 res=round2int(x);  
 9 printf("The result is %d\n", res);  
 10 return 0;  
 11 }  
 12 int round2int(double x){  
 13 int res;  
 14 double temp;  
 15 if(x>0)  
 16 temp = x + 0.5;  
 17 else if(x<0)  
 18 temp = x - 0.5;  
 19 res = (int)temp;  
 20 return res;   
 21 }

Exercise L8-2 (회전변환)

1 #include <stdio.h>  
 2 #include <math.h>  
 3   
 4 void rotate(double \*x, double \*y, double theta);  
 5   
 6 int main(void){  
 7 double x, y, t;  
 8   
 9 printf("Write input data\n");  
 10 printf("x, y, theta : ");  
 11 scanf("%lf %lf %lf", &x, &y, &t);  
 12 t = t \* (M\_PI / 180);  
 13 rotate(&x, &y, t);  
 14 return 0;  
 15 }  
 16   
 17 void rotate(double \*x, double \*y, double theta){  
 18 double X, Y;  
 19 X = (\*x \* cos(theta) - \*y \* sin(theta));  
 20 Y = (\*x \* sin(theta) + \*y \* cos(theta));  
 21 printf("X and Y : %lf %lf\n", X, Y);  
 22 }

Lab5-1 (숫자 반대로 출력)

1 #include <stdio.h>  
 2 int main(void)  
 3 {  
 4 int n, res;  
 5 printf("Put a number n(0<=n<=999 : ");  
 6 scanf("%d", &n);  
 7 res = reverse(n);  
 8 printf("The reserve n is %d\n", res);  
 9 return 0;  
 10 }

1 int reverse(int n) {  
 2 int r = 0;  
 3 while (n != 0) {  
 4 r = r \* 10;  
 5 r = r + n%10;  
 6 n = n/10;  
 7 }  
 8 return r;  
 9 }  
~

Lab5-2 (1+r+r^2+…)

1 #include <stdio.h>  
 2 double GeometricSum(double, int);  
 3   
 4 int main() {  
 5 double r, sum;  
 6 int m;  
 7 printf("This program computes the Geometric Sum of r^n for m+ 1 terms\n");  
 8 printf("Enter r and m : ");  
 9 scanf("%lf %d", &r, &m);  
 10 sum = GeometricSum(r, m);  
 11 printf("The Geometric Sum is %lf\n", sum);  
 12 return 0;  
 13 }

1 double GeometricSum(double r, int m)  
 2 {  
 3 double sum=0, temp=1.0, res;  
 4 int i;  
 5 for(i=0; i<=m; i++){  
 6 sum = sum + temp;  
 7 temp = temp \* r;  
 8 }  
 9 res = sum;  
 10 return res;  
 11 }

Lab5-3 (두 좌표간의 r, theta)

1 #include <stdio.h>  
 2 #include <math.h>  
 3   
 4 void Rect2Pol(double x, double y, double \*r, double \*theta);  
 5   
 6 int main(){  
 7 int x1, x2, y1, y2;  
 8 double r1, r2, t1, t2, r, t;  
 9 printf("Enter (x1, y1) : ");  
 10 scanf("%d %d", &x1, &y1);  
 11 printf("Enter (x2, y2) : ");   
 12 scanf("%d %d", &x2, &y2);  
 13 Rect2Pol(x1, y1, &r1, &t1);  
 14 Rect2Pol(x2, y2, &r2, &t2);  
 15 if(r1 > r2) r = r1 - r2;  
 16 else r = r2 - r1;  
 17 if(t1 > t2) t = t1 - t2;  
 18 else r = t2 - t1;  
 19 printf("The range difference between (%d, %d) and (%d, %d) is %lf\n", x1, y1, x2, y2, r);  
 20 printf("The angle difference between (%d, %d) and (%d, %d) is %lf\n", x1, y1, x2, y2, t);  
 21 }  
 22   
 23 void Rect2Pol(double x, double y, double \*r, double \*theta){  
 24 \*r = sqrt( x\*x + y\*y );  
 25 \*theta = atan( (y / x) );  
 26 }  
 27

Lab5-4 (랜덤 수 출력)

1 #include <stdio.h>  
 2   
 3 int main(void){  
 4 int n, temp;  
 5 printf("This program generates a random number in the range of 0 to n-1\n");  
 6 printf("Enter n : ");  
 7 scanf("%d", &n);  
 8 srand(time(NULL));  
 9 temp = rand() % n;  
 10 printf("A random number in the range of 0 to 999 is %d\n", temp);  
 11 return 0;  
 12 }

Chapter 9. Arrays

Exercise L9-1 (다항식)

1 #include <stdio.h>  
 2 #include <math.h>  
 3 double function(double [], int , int);  
 4 int main(void){  
 5 int n, x, i;  
 6 double a[10]={}, res;  
 7 printf("Put n(1<=n<10) : ");  
 8 scanf("%d", &n);  
 9 printf("Put ai(-10.0<=ai<=10.0) : ");  
 10 for(i=n; i>=0; i--) scanf("%lf", &a[i]);  
 11 printf("Put x(-10.0<=x<=10.0) : "); scanf("%d", &x);  
 12 res = function(a, n, x);  
 13 printf("The function's result is %lf\n", res);  
 14 return 0;  
 15 }  
 16   
 17 double function(double table[], int n, int x)  
 18 {  
 19 double sum;  
 20 int i;  
 21 for(i=0; i <= n; i++){  
 22 sum += pow(x, i) \* table[i];  
 23 }  
 24 return sum;  
 25 }  
~

Exercise L9-2 (파스칼 삼각형)

1 #include <stdio.h>  
 2 int factorial(int x);  
 3 int main(void){  
 4 int array[10][10], n, j, i;  
 5 printf("Enter the row of Pascal's triangle : ");  
 6 scanf("%d", &n);  
 7 for(i=0; i<n; i++){  
 8 for(j=0; j<i; j++)  
 9 array[i][j]=factorial(i)/(factorial(j)\*factorial(i-j));  
 10 printf("\n");  
 11 }  
 12 printf("%d ", array[i][j]);  
 13 return 0;  
 14 }  
 15 int factorial(int x){  
 16 int res=1, y;  
 17 for(y=1; y<=x; y++)  
 18 res = res\*y;  
 19 return res;  
 20 }

Lab 6-2

1 #include <stdio.h>  
 2 int main(){  
 3 int array[10], i, s, e;  
 4 printf("Please enter 10 numbers : ");  
 5 for(i=0; i<10; i++)  
 6 scanf("%d", &array[i]);  
 7 printf("Enter the starting and ending position (1-10) to coty : ");  
 8 scanf("%d %d", &s, &e);  
 9 for(i = s-1; i <= e-1; i++)  
 10 printf("%d ", array[i]);  
 11 array[10] = 0;  
 12 for(i = 9-(e-s); i>0; i--)  
 13 printf("%d ", array[10]);  
 14 putchar('\n');  
 15 return 0;  
 16 }

Lab 6-3 (짝수 세기 – 재귀이용)

1 #include <stdio.h>  
 2 int counteven(int \*numarray, int size);  
 3 int main(){  
 4 int array[10], i, counting, num, \*numarray;  
 5 printf("Please enter 10 numbers : ");  
 6 for(i=0; i<10; i++)  
 7 scanf("%d", array + i);  
 8 for (i=0; i<10; i++)  
 9 printf("%p\n", &array[i]);  
 10 counting = counteven(array, num);  
 11 printf("There are %d even numbers\n", counting);  
 12 return 0;  
 13 }  
 14 int counteven(int \*numarray, int size){  
 15 if (size == 1) {  
 16 if (\*numarray % 2 == 0) return 1;  
 17 else return 0;}  
 18 return counteven(numarray, size / 2) + counteven(numarray + size / 2, size / 2);  
 19 }  
~

Bubble Sorting  
1 #include <stdio.h>  
 2 int main(void){  
 3 int data[100], i, n, step, temp;  
 4 printf("Enter the number of elements to be sorted : ");  
 5 scanf("%d", &n);  
 6 printf("Enter elements : ");  
 7 for(i = 0; i < n; i++)  
 8 scanf("%d", &data[i]);  
 9 for(step = 0; step < n-1; step++)  
 10 for(i = 0; i < n – 1 - step; i++){  
 11 if(data[i] > data[i + 1]){  
 12 temp = data[i];  
 13 data[i] = data[i + 1];  
 14 data[i + 1] = temp;  
 15 }  
 16 }  
 17 printf("In ascending order : ");  
 18 for(i = 0; i < n; i++)  
 19 printf("%-5d", data[i]);  
 20 printf("\n");  
 21 return 0;  
 22 }

Insertion sorting – Ascending

1)  
 1 #include <stdio.h>  
 2 int main(void){  
 3 int data[100], i, j, n, temp;  
 4 printf("Enber number of terms : ");  
 5 scanf("%d", &n);  
 6 printf("Enter elements : ");  
 7 for(i = 0; i < n; i++)  
 8 scanf("%d", &data[i]);  
 9   
 10 for(i = 1; i < n; i++){  
 11 temp = data[i];  
 12 for(j = i; i > 0; j--){  
 13 if(data[j - 1] > temp){  
 14 data[j] = data[j - 1];  
 15 if(j == 1){  
 16 data[j - 1] = temp;  
 17 break;  
 18 }  
 19 }  
 20 else{  
 21 data[j] = temp;  
 22 break;  
 23 }  
 24 }  
 25 }  
 26 printf("In ascending order : ");  
 27 for(i = 0; i < n; i++){  
 28 printf("%d\t", data[i]);  
 29 }  
 30 printf("\n");  
 31 return 0;  
 32 }

2)

1 #include <stdio.h>  
 2 int main(void){  
 3 int data[100], n, temp, i, j;  
 4 printf("Enter number of terms(should be less than 100) : ");  
 5 scanf("%d", &n);  
 6 printf("Enter elements : ");  
 7 for(i = 0; i < n; i++)  
 8 scanf("%d", &data[i]);  
 9 for(i = 1; i < n; i++){  
 10 temp = data[i];  
 11 j = i - 1;  
 12 while(temp<data[j] && j>=0){  
 13 data[j+1] = data[j];  
 14 --j;  
 15 }  
 16 data[j+1] = temp;  
 17 }  
 18 printf("In ascending order : ");  
 19 for(i = 0; i < n; i++)  
 20 printf("%d\t", data[i]);  
 21 printf("\n");  
 22 return 0;  
 23 }  
~

Insertion sorting – Descending

1 #include <stdio.h>  
 2 int main(void){  
 3 int data[100], i, j, n, k, hold;  
 4 printf("Enter number of terms(should be less than 100) : ");  
 5 scanf("%d", &n);  
 6 printf("Enter elements : ");  
 7 for(i = 0; i < n; i++)  
 8 scanf("%d", &data[i]);  
 9 for(i = 1; i <n; i++)  
 10 {  
 11 for(j = 0; j < i; j++)  
 12 if(data[j] < data[i]){  
 13 hold = data[i];  
 14 k = i;  
 15 while(k != j){  
 16 data[k] = data[k-1];  
 17 --k;  
 18   
 19 }  
 20 data[j] = hold;  
 21 }  
 22 }  
 23 printf("In descending Order : ");  
 24 for(i = 0; i < n; i++)  
 25 printf("%d ", data[i]);  
 26 putchar('\n');  
 27 return 0;  
 28 }

Selection sorting – Ascending

1 #include <stdio.h>  
 2 int main(void){  
 3 int data[100], i, n, steps, temp;  
 4 printf("Enter the number of elements to be sorted : ");  
 5 scanf("%d", &n);  
 6 printf("Enter the elements : ");  
 7 for(i = 0; i < n; i++){  
 8 scanf("%d", &data[i]);  
 9 }  
 10 for(steps = 0; steps < n; steps++)  
 11 for(i = steps + 1; i <n; i++){  
 12 if(data[steps] > data[i]){  
 13 temp = data[steps];  
 14 data[steps] = data[i];  
 15 data[i] = temp;  
 16 }  
 17 }  
 18 printf("In ascending order : ");  
 19 for(i = 0; i < n; i++)  
 20 printf("%d ", data[i]);  
 21 putchar('\n');  
 22 return 0;  
 23 }

Binary Searching using “Recursion”

1 #include <stdio.h>  
 2   
 3 int RecursiveBinarySearching(int arr[], int low, int high, int element);  
 4   
 5 int main(){  
 6 int count, element, limit, arr[50], position;  
 7 printf("Enter the limit of elements in array:\t");  
 8 scanf("%d", &limit);  
 9 printf("\nEnter %d elements in array: \n", limit);  
 10   
 11 for(count = 0; count < limit; count++){  
 12 scanf("%d", &arr[count]);  
 13 }  
 14 printf("\nEnter element to search:\t");  
 15 scanf("%d", &element);  
 16 position = RecursiveBinarySearching(arr, 0, limit - 1, element);  
 17   
 18 if(position == -1){  
 19 printf("\nEnter %d Not Found\n", element);  
 20 }  
 21 else {  
 22 printf("\nEnter %d Found at Position %d\n", element, position + 1);  
 23 }  
 24 return 0;  
 25 }  
 26   
 27 int RecursiveBinarySearching(int arr[], int low, int high, int element){  
 28 int middle;  
 29 if (low > high) {  
 30 return -1;  
 31 }  
 32 middle = (low + high) / 2;  
 33 if(element > arr[middle]) RecursiveBinarySearching(arr, middle + 1, high, element);  
 34 else if(element < arr[middle]) RecursiveBinarySearching(arr, low, middle - 1, element);  
 35 else { return middle;

}

MERGE SORT

1. */\**
2. *\* C Program to Input Few Numbers & Perform Merge Sort on them using Recursion*
3. *\*/*
5. #include <stdio.h>
7. void mergeSort(int [], int, int, int);
8. void partition(int [],int, int);
10. int main()
11. {
12. int list[50];
13. int i, size;
15. printf("Enter total number of elements:");
16. scanf("%d", &size);
17. printf("Enter the elements:**\n**");
18. for(i = 0; i < size; i++)
19. {
20. scanf("%d", &list[i]);
21. }
22. partition(list, 0, size - 1);
23. printf("After merge sort:**\n**");
24. for(i = 0;i < size; i++)
25. {
26. printf("%d ",list[i]);
27. }
29. return 0;
30. }
32. void partition(int list[],int low,int high)
33. {
34. int mid;
36. if(low < high)
37. {
38. mid = (low + high) / 2;
39. partition(list, low, mid);
40. partition(list, mid + 1, high);
41. mergeSort(list, low, mid, high);
42. }
43. }
45. void mergeSort(int list[],int low,int mid,int high)
46. {
47. int i, mi, k, lo, temp[50];
49. lo = low;
50. i = low;
51. mi = mid + 1;
52. while ((lo <= mid) && (mi <= high))
53. {
54. if (list[lo] <= list[mi])
55. {
56. temp[i] = list[lo];
57. lo++;
58. }
59. else
60. {
61. temp[i] = list[mi];
62. mi++;
63. }
64. i++;
65. }
66. if (lo > mid)
67. {
68. for (k = mi; k <= high; k++)
69. {
70. temp[i] = list[k];
71. i++;
72. }
73. }
74. else
75. {
76. for (k = lo; k <= mid; k++)
77. {
78. temp[i] = list[k];
79. i++;
80. }
81. }
83. for (k = low; k <= high; k++)
84. {
85. list[k] = temp[k];
86. }
87. }
88. */\**
89. *\* C program to read N names, store them in the form of an array*
90. *\* and sort them in alphabetical order. Output the given names and*
91. *\* the sorted names in two columns side by side.*
92. *\*/*
94. #include <stdio.h>
95. #include <string.h>
96. void main()
97. {
99. char name[10][8], tname[10][8], temp[8];
100. int i, j, n;
102. printf("Enter the value of n **\n**");
103. scanf("%d", &n);
104. printf("Enter %d names n", \n);
106. for (i = 0; i < n; i++)
107. {
108. scanf("%s", name[i]);
109. strcpy(tname[i], name[i]);
110. }
112. for (i = 0; i < n - 1 ; i++)
113. {
114. for (j = i + 1; j < n; j++)
115. {
116. if (strcmp(name[i], name[j]) > 0)
117. {
118. strcpy(temp, name[i]);
119. strcpy(name[i], name[j]);
120. strcpy(name[j], temp);
121. }
122. }
123. }
125. printf("**\n**----------------------------------------**\n**");
126. printf("Input NamestSorted names**\n**");
127. printf("------------------------------------------**\n**");
129. for (i = 0; i < n; i++)
130. {
131. printf("%s**\t\t**%s**\n**", tname[i], name[i]);
132. }
134. printf("------------------------------------------**\n**");
136. }

Chapter 10. Character strings & string function

1 #include <stdio.h>  
 2 #include <string.h>  
 3   
 4 #define MAX 20  
 5   
 6 int main(void){  
 7 char name[MAX];  
 8 printf("Hi, what is your name?\n");

9 fgets(name, MAX, stdin);

10 name[strcspn(name, "\n")] = '\0';

11 printf("Nice name, %s.\n", name);  
 12 return 0;  
 13 }

알파벳 빈도수

#include <stdio.h>

int main()

{

char str[1000], ch;

int i, frequency = 0;

printf("Enter a string: ");

gets(str);

printf("Enter a character to find the frequency: ");

scanf("%c",&ch);

for(i = 0; str[i] != '\0'; ++i)

{

if(ch == str[i])

++frequency;

}

printf("Frequency of %c = %d", ch, frequency);

return 0;

}

Enter a string: This website is awesome.

Enter a character to find the frequency: e

Frequency of e = 4

알파벳 순서대로 출력 (사전)

#include<stdio.h>

#include <string.h>

int main()

{

int i, j;

char str[10][50], temp[50];

printf("Enter 10 words:\n");

for(i=0; i<10; ++i)

scanf("%s[^\n]",str[i]);

for(i=0; i<9; ++i)

for(j=i+1; j<10 ; ++j)

{

if(strcmp(str[i], str[j])>0)

{

strcpy(temp, str[i]);

strcpy(str[i], str[j]);

strcpy(str[j], temp);

}

}

printf("\nIn lexicographical order: \n");

for(i=0; i<10; ++i)

{

puts(str[i]);

}

return 0;

}

Reverse

1 #include <stdio.h>  
 2 #include <string.h>  
 3 void reverse(char \*);  
 4 int main(){  
 5 char line[132];  
 6 fgets(line, 132, stdin);  
 7 line[strcspn(line, "\n")] = '\0';  
 8 while(strcmp(line, "ZZZ") != 0){  
 9 reverse(line);  
 10 printf("%s\n", line);  
 11 fgets(line, 132, stdin);  
 12 line[strcspn(line, "\n")] = '\0';  
 13 }  
 14 }  
 15 void reverse(char \*s){  
 16 char c, \*end;  
 17   
 18 end = s + strlen(s) - 1;  
 19 while(s < end){  
 20 c = \*s;  
 21 \*s++ = \*end;  
 22 \*end-- = c;  
 23 }  
 24 }

Lab7-1

1 #include <stdio.h>  
 2 #include <string.h>  
 3 int main(){  
 4 char s1[100], s2[100], s3[100];  
 5 printf("Please enter the first string : ");  
 6 scanf("%s", s1);  
 7 printf("Please enter the second string : ");  
 8 scanf("%s", s2);  
 9 if (strcmp(s1, s2) != 0){  
 10 strcpy(s3, s1);  
 11 strcat(s3, s2);  
 12 }  
 13 else  
 14 strcpy(s3, s1);  
 15 printf("The final string is %s\n", s3);  
 16 return 0;  
 17 }

Lab7-2

1 #include <stdio.h>  
 2 #include <stdlib.h>  
 3 #include <string.h>  
 4 #include <ctype.h>  
 5   
 6 int main(){  
 7 char array[100];  
 8 char buffer[100] = {0. };  
 9 int cnt = 0, i = 0, len, x;  
 10 printf("Please enter your annual salary : ");  
 11 scanf("%s", array);  
 12 len = strlen(array);  
 13 while (i < len){

/\* array의 길이만큼 각각의 문자를 조사 \*/  
 14 if(isdigit(array[i]) || array[i] == '.'){

/\* array안에 문자가 숫자이거나 .이면 buffer에 대입

15 buffer[cnt++] = array[i];  
 16 }  
 17 i++;  
 18 }  
 19 x = atoi(buffer); /\* buffer안에 문자를 숫자로 바꿈 \*/  
 20 printf("The integer value of your salary is %d\n", x);  
 21 return 0;  
 22 }

Lab7-3 (matrix 입출력)

1 #include <stdio.h>  
 2 #include <stdlib.h>  
 3 int main(){  
 4 int m, n, i, j;  
 5 double \*\*ptr;  
 6 printf("Please enter the matrix size (m\*n) : ");  
 7 scanf("%d %d", &m, &n);  
 8 ptr = (double\*\*)malloc(sizeof(double\*)\*m);  
 9 for(i = 0; i < m; i++)  
 10 ptr[i] = (double\*)malloc(sizeof(double)\*n);

11 printf("Please enter the %d \* %d elements : ", m, n);

12 for(i = 0; i < m; i++)  
 13 for(j = 0; j < n; j++)  
 14 scanf("%ld", ptr);

15 printf("The matrix you entered is : \n");

16 for(i = 0; i < m; i++){  
 17 for(j = 0; j < n; j++){  
 18 printf("%ld ", ptr[i][j]);  
 19 }  
 20 printf("\n");  
 21 }

22 return 0;  
 23   
 24 }

Lab7-3

1 #include <stdio.h>  
 1 #include <stdio.h>  
 2   
 3 int main(void){  
 5 printf("Enter rows and column for first matrix : ");  
 6 scanf("%d %d", &r1, &c1);  
 7   
 8 printf("Enter rows and column for second matrix : ");  
 9 scanf("%d %d", &r2, &c2);  
 10   
 12 while (c1 != r2){  
 13 printf("Error!\n");  
 14 }  
 15 //Storing elements of first matrix  
 16 printf("Enter elements of matrix 1 : \n");  
 17 for(i = 0; i < r1; i++){  
 18 for(j = 0; j < c1; j++)  
 19 {  
 20 printf("Enter elements a%d%d: ", i+1, j+1);  
 21 scanf("%d", &a[i][j]);  
 22 }  
 23 //Storing elements of second matrix  
 24 printf("Enter elements of matrix 2:\n");  
 25 for(i = 0; i < r2; i++)  
 26 for(j = 0; j <c2; j++){  
 27 printf("Enter elements b%d%d: ", i+1, j+1);  
 28 scanf("%d", &b[i][j]);  
 29 }  
 30   
 31 //Multiplying matrices a and b and  
 32 //storing result in result matrix  
 33 for(i = 0; i <r1; i++)  
 34 for(j = 0; j < c2; j++)  
 35 for(k = 0; k < c1; k++)  
 36 result[i][j] += a[i][k] \* b[k][j];  
 37   
 38 //Displaying the result  
 39 printf("\nOutput Matrix:\n");  
 40 for(i = 0; i <r1; i++)  
 41 for(j = 0; j <c2; j++)  
 42 {  
 43 printf("%d\t", result[i][j]);  
 44 if(j == c2 -1) printf("\n\n");  
 45 }   
 46 return 0;

Chapter 11. Structures

Lab 8-1

1 #include <stdio.h>  
 2   
 3 struct complex{  
 4 double Re;  
 5 double Im;  
 6 };  
 7   
 8 int main(void){  
 9 struct complex z;  
 10 printf("Enter the real and imaginary parts of a Complex Num ber : ");  
 11 scanf("%lf %lf", &z.Re, &z.Im);  
 12 printf("The Complex Number you entered is %lf-%lfi\n", z.Re , z.Im);  
 13 return 0;  
 14 }  
~

Lab 8-2

1 #include <stdio.h>  
 2 #include <math.h>  
 3   
 4 struct complex{  
 5 double Re;  
 6 double Im;  
 7 };  
 8   
 9 void print\_result(struct complex \*p);  
 10   
 11 int main(void){  
 12 struct complex z;  
 13 printf("Enter the rectangular coordinates of Complex Number z1 : ");  
 14 scanf("%lf %lf", &z.Re, &z.Im);  
 15 printf("z1 = %lf+%lf\n", z.Re, z.Im);  
 16 print\_result(&z);  
 17 return 0;  
 18 }  
 19   
 20 void print\_result(struct complex \*p){  
 21 double Mag;  
 22 double Arg;  
 23 Mag = sqrt((p -> Re)\*(p -> Re) + (p -> Im)\*(p -> Im));  
 24 Arg = atan((p -> Im) / (p -> Re));  
 25   
 26 printf("z2 = (%lf, %lf)\n", Mag, Arg);  
 27 }

Lab 8-3

1 #include <stdio.h>  
 2 #include <math.h>  
 3   
 4 typedef struct{  
 5 double mag[10];  
 6 double ang[10];  
 7 }complex;  
 8   
 9 int main(void){  
 10 double num;  
 11 complex list[10];  
 12 int i;  
 13 printf("Enter 10 Complex Numbers of the polar form : ");  
 14 for(i = 0; i < 10; i++){  
 15 scanf("%lf %lf", &list[i].mag[i], &list[i].ang[i]);  
 16 }  
 17 printf("Enter a magnitude search key : ");  
 18 scanf("%lf", &num);  
 19 for( i = 0; i < 10; i++){  
 20 if(fabs(num)-0.01 <= list[i].mag[i] && fabs(num)+0.01 > = list[i].mag[i])  
 21 printf("The Complex Number (%d, %lf) is found at po sition %d\n", i, list[i].mag[i], i);  
 22 }  
 23 return 0;  
 24 }

Lab8-4

1 #include <stdio.h>  
 2 #include <math.h>  
 3   
 4 union complex{  
 5   
 6 struct R{  
 7 double Re;  
 8 double Im;  
 9 }rect;  
 10   
 11 struct P{  
 12 double mag;  
 13 double ang;  
 14 }polar;  
 15   
 16 };  
 17   
 18 void convert\_Rect(double p, double q);  
 19 void convert\_Polar(double r, double s);  
 20   
 21 int main(void){  
 22 union complex z;  
 23 char ch;  
 24 printf("Please enter complex number representation form ([R ]ect, [P]olar) : ");  
 25 scanf("%c", &ch);  
 26 if(ch == 'r'){  
 27 printf("Please enter real and imaginary parts : ");  
 28 scanf("%lf %lf", &z.rect.Re, &z.rect.Im);  
 29 convert\_Rect(z.rect.Re, z.rect.Im);  
 30 }  
 31 if(ch == 'p'){  
 32 printf("Please enter magnitude and argument : ");



#define \_CRT\_SECURE\_NO\_WARNINGS // fopen 보안 경고로 인한 컴파일 에러 방지

#include <stdio.h> // fopen, feof, fread, fclose 함수가 선언된 헤더 파일

#include <string.h> // strlen, memset 함수가 선언된 헤더 파일

int main()

{

char buffer[5] = { 0, }; // 문자열 데이터 4바이트 NULL 1바이트. 4 + 1 = 5

int count = 0;

int total = 0;

FILE \*fp = fopen("hello.txt", "r"); // hello.txt 파일을 읽기 모드(r)로 열기.

// 파일 포인터를 반환

while (feof(fp) == 0) // 파일 포인터가 파일의 끝이 아닐 때 계속 반복

{

count = fread(buffer, sizeof(char), 4, fp); // 1바이트씩 4번(4바이트) 읽기

printf("%s", buffer); // 읽은 내용 출력

memset(buffer, 0, 5); // 버퍼를 0으로 초기화

total += count; // 읽은 크기 누적

}

printf("\ntotal: %d\n", total); // total: 13: 파일을 읽은 전체 크기 출력

fclose(fp); // 파일 포인터 닫기

return 0;

}

#define \_CRT\_SECURE\_NO\_WARNINGS // fopen 보안 경고로 인한 컴파일 에러 방지

#include <stdio.h> // fopen, fwrite, fclose 함수가 선언된 헤더 파일

#pragma pack(push, 1) // 1바이트 크기로 정렬

struct Data {

short num1; // 2바이트

short num2; // 2바이트

short num3; // 2바이트

short num4; // 2바이트

};

#pragma pack(pop) // 정렬 설정을 이전 상태(기본값)로 되돌림

int main()

{

struct Data d1;

d1.num1 = 100;

d1.num2 = 200;

d1.num3 = 300;

d1.num4 = 400;

FILE \*fp = fopen("data.bin", "wb"); // 파일을 쓰기/바이너리 모드(wb)로 열기

fwrite(&d1, sizeof(d1), 1, fp); // 구조체의 내용을 파일에 저장

fclose(fp); // 파일 포인터 닫기

return 0;

}