**Homework 9**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 프로그래밍 언어 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.05.11.화 |

1. **구조체를 사용하여 함수에 인수를 전달하고, 결과값을 반환시키는 기능을 예를 들어 설명하라.**

* call-by-value  
  구조체를 함수의 인수로 전달하는 경우 그 구조체의 복사본이 함수로 전달된다.  
  따라서, 구조체의 크기가 크면 그만큼 긴 시간과 큰 메모리가 소요된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* call-by-pointer  
  구조체의 포인터를 함수의 인수로 전달하는 경우 전자와 다르게 시간과 공간을 절약할 수 있다. 하지만 원본 훼손의 가능성이 있으므로 const로 설정하여 변경을 방지해야한다. 구조체 포인터를 인수로 전달받은 후, 표준 입력 장치로부터 세부 데이터를 입력받아 구조체의 각 속성값으로 설정한다. 구조체 포인터를 통해 해당 구조체의 데이터를 직접 변경할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **Byte Ordering이란 무엇인가에 대하여 설명하고, 인터넷을 통한 정보 전송에서 바이트 저장순서를 고려하는 이유는 무엇인가?**

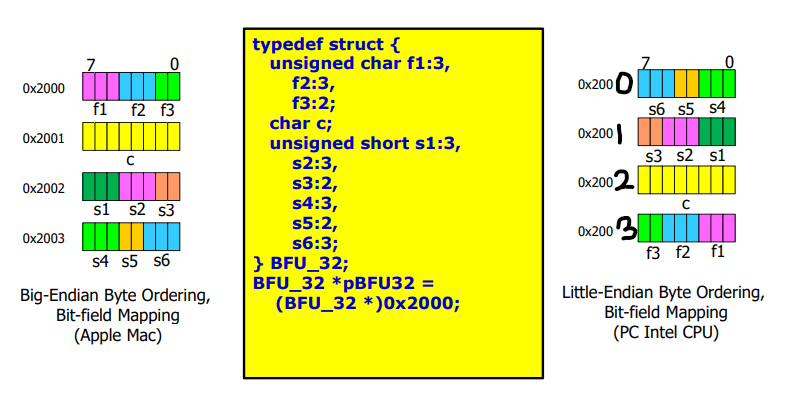
컴퓨터에서 사용되는 CPU 종류에 따라 여러 바이트로 표현되는 데이터를 저장할 때, 바이트 저장 순서가 서로 다르다.

* Big Endian 체계  
  기준이 되는 시작 주소에 MSB인 Big End가 저장된다.
* Little Endian 체계  
  기준이 되는 시작 주소에 LSB인 Little End가 저장된다.

하나의 컴퓨터 내에서만 데이터를 저장하고, 읽어 사용하는 경우 문제가 없다. 하지만 인터넷을 통하여 다른 컴퓨터를 데이터를 전달하는 경우, 각 송신측과 수신측 컴퓨터의 바이트 저장 순서 체계가 다른 경우, 문제가 발생된다. 이는 인터넷으로 데이터를 전송할 때, 항상 Big Endian 체계로 변환시켜서 전송하면 해결되는 문제다.

1. **비트 필드 구조체란 무엇인가, 그리고 왜 필요한가에 대하여 설명하고, Big Endian CPU와 Little Endian CPU가 각각 저장하는 차이점을 예를 들어 설명하라.**

구조체 멤버의 자료형 내부에서 원하는 크기의 비트를 따로 할당해 주는 것을 비트 필드 구조체라고 한다. 이는 메모리 공간을 효율적으로 활용하기 위해 사용한다.



1. **복소수 계산을 위한 구조체 정의 (Struct Cmplx)**

/\* Cmplx.h \*/

#ifndef CMPLX\_H

#define CMPLX\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

typedef struct {

double real;

double imag; // imaginary

} Cmplx;

Cmplx inputCmplx();

double compareCmplx(const Cmplx c1, const Cmplx c2);

void printCmplx(const Cmplx c);

Cmplx cmplxAdd(const Cmplx c1, const Cmplx c2);

Cmplx cmplxSub(const Cmplx c1, const Cmplx c2);

Cmplx cmplxMul(const Cmplx c1, const Cmplx c2);

Cmplx cmplxDiv(const Cmplx c1, const Cmplx c2);

void \_sortCmplx(Cmplx cmplxs[], int start, int end);

void sortCmplx(Cmplx cmplxs[], int size);

#endif

/\* main\_cmplx.cpp (1) \*/

/\*

\* \* 파일명 : main\_cmplx.cpp

\* 목적 및 기본 기능

\* 복소수의 사칙연산과 정렬

\* 작성자 : 김주환(21812158)

\* 작성일 : 2021년 5월 11일

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>.

#include <stdlib.h>

#include <math.h>

#include "Cmplx.h"

void main() {

Cmplx cmplxs[7];

cmplxs[0] = inputCmplx();

cmplxs[1] = inputCmplx();

printf("cmplxs[0] = "); printCmplx(cmplxs[0]); printf("\n");

printf("cmplxs[1] = "); printCmplx(cmplxs[1]); printf("\n");

cmplxs[2] = cmplxAdd(cmplxs[0], cmplxs[1]);

printf("cmplxs[2] = cmplxs[0] + cmplxs[1] = \n ");

printCmplx(cmplxs[0]); printf(" + ");

printCmplx(cmplxs[1]); printf(" = ");

printCmplx(cmplxs[2]); printf("\n");

cmplxs[3] = cmplxSub(cmplxs[0], cmplxs[1]);

printf("cmplxs[3] = cmplxs[0] - cmplxs[1] = \n ");

printCmplx(cmplxs[0]); printf(" - ");

printCmplx(cmplxs[1]); printf(" = ");

printCmplx(cmplxs[3]); printf("\n");

cmplxs[4] = cmplxMul(cmplxs[0], cmplxs[1]);

printf("cmplxs[4] = cmplxs[0] \* cmplxs[1] = \n ");

printCmplx(cmplxs[0]); printf(" \* ");

printCmplx(cmplxs[1]); printf(" = ");

printCmplx(cmplxs[4]); printf("\n");

cmplxs[5] = cmplxDiv(cmplxs[0], cmplxs[1]);

printf("cmplxs[5] = cmplxs[0] / cmplxs[1] = \n ");

printCmplx(cmplxs[0]); printf(" / ");

printCmplx(cmplxs[1]); printf(" = ");

printCmplx(cmplxs[5]); printf("\n");

cmplxs[6] = cmplxMul(cmplxs[1], cmplxs[5]);

printf("cmplxs[6] = cmplxs[1] \* cmplxs[5] = \n ");

printCmplx(cmplxs[1]); printf(" \* ");

printCmplx(cmplxs[5]); printf(" = ");

printCmplx(cmplxs[6]); printf("\n");

printf("Before sorting complexs: \n");

for (int i = 0; i < 7; i++) {

printf("cmplxs[%d] = ", i); printCmplx(cmplxs[i]);

printf("\n");

}

sortCmplx(cmplxs, 7);

printf("Sorted complexs: \n");

for (int i = 0; i < 7; i++) {

printf("cmplxs[%d] = ", i); printCmplx(cmplxs[i]);

printf("\n");

}

}

/\* Cmplx.cpp(2) \*/

Cmplx cmplxDiv(const Cmplx c1, const Cmplx c2) {

Cmplx c\_div;

c\_div.real = (c1.real \* c2.real + c1.imag \* c2.imag) / (pow(c2.real, 2) + pow(c2.imag, 2));

c\_div.imag = (c1.imag \* c2.real - c1.real \* c2.imag) / (pow(c2.real, 2) + pow(c2.imag, 2));

return c\_div;

}

void \_sortCmplx(Cmplx cmplxs[], int start, int end) {

int pivot = (end + start) / 2;

int j = start, key = end;

Cmplx temp;

if (end <= start)

return;

temp = cmplxs[pivot];

cmplxs[pivot] = cmplxs[end];

cmplxs[end] = temp;

for (int i = start; i < end; i++)

if (compareCmplx(cmplxs[i], cmplxs[key]) <= 0)

{

temp = cmplxs[i];

cmplxs[i] = cmplxs[j];

cmplxs[j] = temp;

j++;

}

temp = cmplxs[j];

cmplxs[j] = cmplxs[key];

cmplxs[key] = temp;

\_sortCmplx(cmplxs, start, j - 1);

\_sortCmplx(cmplxs, j + 1, end);

}

void sortCmplx(Cmplx cmplxs[], int size)

{

\_sortCmplx(cmplxs, 0, size - 1);

}

#endif

/\* Cmplx.cpp \*/

#ifndef CMPLX\_H

#define CMPLX\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include "Cmplx.h"

typedef struct {

double real;

double imag; // imaginary

} Cmplx;

Cmplx inputCmplx() {

Cmplx create;

double temp1, temp2;

printf("Input complex number (rrr.ddd iii.ddd) : ");

scanf("%lf %lf", &temp1, &temp2);

create.real = temp1;

create.imag = temp2;

return create;

}

double compareCmplx(const Cmplx c1, const Cmplx c2) {

double mag1, mag2;

mag1 = (double)sqrt(pow(c1.real, 2) + pow(c1.imag, 2));

mag2 = (double)sqrt(pow(c2.real, 2) + pow(c2.imag, 2));

return mag1 - mag2;

}

void printCmplx(const Cmplx c) {

double mag;

mag = (double)sqrt(pow(c.real, 2) + pow(c.imag, 2));

if (c.imag >= 0) {

printf("(%3.3f + j%3.3f (magnitude %3.3f))", c.real, c.imag, mag);

}

else {

printf("(%3.3f - j%3.3f (magnitude %3.3f))", c.real, -c.imag, mag);

}

}

Cmplx cmplxAdd(const Cmplx c1, const Cmplx c2) {

Cmplx c\_add;

c\_add.real = c1.real + c2.real;

c\_add.imag = c1.imag + c2.imag;

return c\_add;

}

Cmplx cmplxSub(const Cmplx c1, const Cmplx c2) {

Cmplx c\_sub;

c\_sub.real = c1.real - c2.real;

c\_sub.imag = c1.imag - c2.imag;

return c\_sub;

}

Cmplx cmplxMul(const Cmplx c1, const Cmplx c2) {

Cmplx c\_mul;

c\_mul.real = c1.real \* c2.real - c1.imag \* c2.imag;

c\_mul.imag = c1.real \* c2.imag + c1.imag \* c2.real;

return c\_mul;

}

**텍스트이(가) 표시된 사진

자동 생성된 설명**

1. **구조체 Time을 사용하는 Clock system**

/\* Time.h \*/

#ifndef TIME\_H

#define TIME\_H

#include <stdio.h>

#include <stdlib.h>

typedef struct

{

int hour;

int min;

int sec;

} Time;

Time inputTime();

Time initTime(int h, int m, int s);

void printTime(const Time\* t);

void incrementTime(Time\* t, int incr\_sec);

int elapsedSeconds(const Time\* t);

int compareTime(const Time\* t1, const Time\* t2);

void selectSortTime(Time\* times, int size);

#endif

/\* main\_time.cpp (1) \*/

/\*

\* \* 파일명 : main\_time.cpp

\* 목적 및 기본 기능

\* 시간의 경과와 정렬

\* 작성자 : 김주환(21812158)

\* 작성일 : 2021년 5월 11일

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include "Time.h"

#define NUM\_TIMES 5

void main() {

Time t1, t2;

Time times[NUM\_TIMES];

int incr\_secs, diff\_sec;

t1 = t2 = inputTime();

printf("Input time t1 = "); printTime(&t1); printf("\n");

printf("input seconds to increment : ");

scanf("%d", &incr\_secs);

incrementTime(&t2, incr\_secs);

printf("After incrementing %d secs, t2 = ", incr\_secs); printTime(&t2); printf("\n");

diff\_sec = compareTime(&t1, &t2);

printf("Difference between t1 and t2 is %d secs\n", diff\_sec);

times[0] = initTime(23, 59, 59);

times[1] = initTime(9, 0, 5);

times[2] = initTime(13, 30, 0);

times[3] = initTime(3, 59, 59);

times[4] = initTime(0, 0, 0);

printf("\nBefore sorting times : \n");

for (int i = 0; i < NUM\_TIMES; i++) {

printf("times[%d] = ", i); printTime(&times[i]); printf("\n");

}

selectSortTime(times, NUM\_TIMES);

printf("After selection sorting of times : \n");

for (int i = 0; i < NUM\_TIMES; i++) {

printf("times[%d] = ", i); printTime(&times[i]); printf("\n");

}

}

int compareTime(const Time\* t1, const Time\* t2) {

int f, s;

int diff = 0;

f = t1->hour \* 3600 + t1->min \* 60 + t1->sec;

s = t2->hour \* 3600 + t2->min \* 60 + t2->sec;

if (diff < 0)

diff = 3600 \* 24 - f + s;

return s - f;

}

void selectSortTime(Time\* times, int size) {

int\* secA, sec, min, temp;

Time temp\_st;

secA = (int\*)calloc(size, sizeof(int));

if (secA == NULL)

exit(-1);

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

{

sec = 3600 \* times[i].hour + 60 \* times[i].min + times[i].sec;

secA[i] = sec;

}

for (int i = 0; i < size - 1; i++)

{

min = i;

for (int j = i + 1; j < size; j++)

{

if (secA[j] < secA[min])

min = j;

}

if (min != i)

{

temp\_st = times[i];

times[i] = times[min];

times[min] = temp\_st;

temp = secA[i];

secA[i] = secA[min];

secA[min] = temp;

}

}

free(secA);

}

#endif

/\* Time.h \*/

#ifndef TIME\_H

#define TIME\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include "Time.h"

typedef struct {

int hour;

int min;

int sec;

} Time;

Time inputTime() {

Time create;

create.hour = 0;

create.min = 0;

create.sec = 0;

while (1) {

printf("input hour minute sec : ");

scanf\_s("%d %d %d", &create.hour, &create.min, &create.sec);

if ((create.hour < 24) && (create.hour >= 0) &&

(create.min < 60) && (create.min >= 0) &&

(create.sec < 60) && (create.sec >= 0))

break;

}

return create;

}

Time initTime(int h, int m, int s) {

Time create;

create.hour = h;

create.min = m;

create.sec = s;

return create;

}

void printTime(const Time\* t) {

printf("(%02d:%02d:%02d)\n", t->hour, t->min, t->sec);

}

void incrementTime(Time\* t, int incr\_sec) {

for (int i = 1; i <= incr\_sec; i++) {

t->sec += 1;

if (t->sec == 60) {

t->sec = 0;

t->min += 1;

if (t->min == 60) {

t->min = 0;

t->hour += 1;

if (t->hour == 24)

t->hour = 0;

}

}

}

}

**텍스트이(가) 표시된 사진

자동 생성된 설명**