**연습문제에 대한 필자의 답안**

**제1장**

**1.**

0xFFFF

**2.**

11000000(2)

**3.**

110100011111(2)

**4.**

XOR

**제2장**

**1.**

#include <stdio.h>

void main()

{

printf("최호성\n");

}

**2.**

#include <stdio.h>

void main()

{

printf("최호성\n010-1234-5678\n");

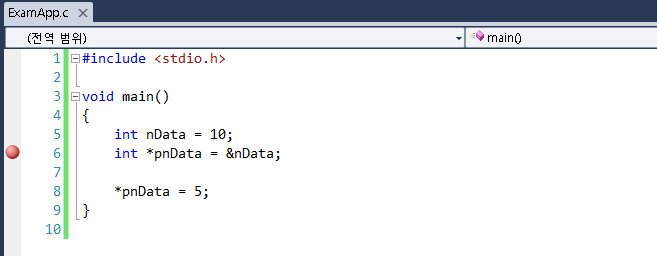
}

**3.**

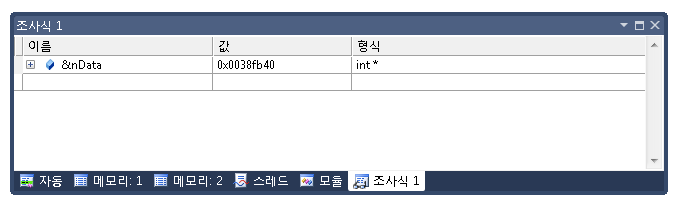
컴파일과 링크. 두 과정을 하나로 묶어서 '빌드'라고 합니다.

**제3장**

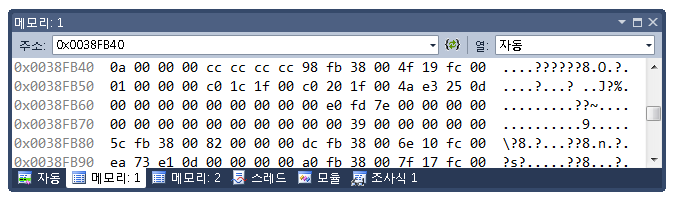
**1.** 위치 중단점을 설정하는 문제입니다. 다음 그림과 같이 6행에 중단점이 설정되어야 정답입니다.



**2.** 조사식 창에 직접 값을 입력하면 다음 그림과 같은 결과를 볼 수 있습니다. 그러나 주소(값)는 시스템 마다 다를 수 있습니다.



**3.** 2번 문제에서 확인한 nData 변수의 주솟값이 들어 있어야 합니다. 단, 주소는 시스템마다 다를 수 있습니다.



**제4장**

**1.** getchar() 함수를 처음 호출했을 때는 문자열을 입력 받지만, 이후로는 버퍼에서 한 글자를 읽어와 반환하므로 단순하게 2회 호출하면 두 번째 문자를 얻을 수 있습니다.

#include <stdio.h>

void main()

{

getchar();

printf("%c\n", getchar());

}

**2.** 문자열의 최대 길이가 32미만이므로 최소 32바이트 이상의 공간이 필요하다고 할 수 있습니다. 마지막에 NULL이 저장될 공간도 고려해야 하기 때문입니다.

#include <stdio.h>

void main()

{

char szBuffer[32] = {0};

gets(szBuffer);

printf("%s\n", szBuffer);

}

**3.** %p 형식 문자를 이용하면 주소를 쉽게 출력할 수 있습니다.

#include <stdio.h>

void main()

{

char szBuffer[32] = {0};

gets(szBuffer);

printf("%p %s\n", szBuffer, szBuffer);

}

**4.** 아스키 코드는 숫자와 문자를 단순히 맵핑(Mapping)한 표입니다. 따라서 문자를 숫자 형식으로 출력할 수 있습니다.

<소스코드>

#include <stdio.h>

#include <conio.h>

void main()

{

char ch = 0;

ch = getch();

printf("%d\n", ch);

}

**5.** 형식문자 %o, %x혹은 %X에 대한 문제입니다.

#include <stdio.h>

void main()

{

int nData = 0;

printf("Input number : ");

scanf("%d", &nData);

printf("%o %X\n", nData, nData);

}

**6.** 아스키 코드 표를 보면 영문 대문자와 소문자는 32만큼의 값 차이가 있음을 알 수 있습니다. 이러한 특징을 이용하여 직접 계산할 수 있습니다.

#include <stdio.h>

#include <conio.h>

void main()

{

char ch = 0;

ch = getch();

printf("%c\n", ch - 32);

}

**7.** %f 형식 문자 사용에 대한 문제입니다.

#include <stdio.h>

void main()

{

float fData1 = 0.0f, fData2 = 0.0f;

scanf("%f %f", &fData1, &fData2);

printf("%.2f\n", (fData1 + fData2) / 2);

}

**8.** %d와 %u 형식 문자의 특징을 이해하고 있는지 확인하는 문제입니다. nData 변수에 저장된 값이 달라지는 것은 아니지만 출력하는 형식에 따라서 전혀 다른 의미를 가질 수도 있습니다.

#include <stdio.h>

void main()

{

int nData = 0;

scanf("%d", &nData);

printf("%u\n%X\n", nData, nData);

}

**제5장**

**1.** 기본 사칙 연산자의 사용 방법에 대한 문제입니다.

#include <stdio.h>

void main()

{

int x = 0, y = 0;

scanf("%d %d", &x, &y);

printf("%d, %d, %d, %d\n",

x + y,

x - y,

x \* y,

x / y);

}

**2.** 나머지 연산자에 대한 문제입니다.

#include <stdio.h>

void main()

{

int nData = 0;

scanf("%d", &nData);

printf("%d\n", nData % 5);

}

**3.** 비트 연산 및 마스크 연산에 대한 문제입니다.

#include <stdio.h>

void main()

{

int nData = 0;

scanf("%d", &nData);

nData = nData & 0xFFFFFFF8;

printf("%d\n", nData);

}

**4.** 이 문제는 특정 위치의 한 비트만 골라서 제어하는 방법에 대한 문제입니다. 실제로 사용할 가능성이 많은 중요한 비트 연산 문제입니다. 특히 윈도우 프로그래밍에 대해서 공부할 예정이라면 이와 같은 코드에 대해서 명확히 이해하고 있어야 합니다.

#include <stdio.h>

void main()

{

int nData = 0;

scanf("%d", &nData);

nData &= ~0x00000008;

printf("%d\n", nData);

}

**5.** '-'가 2항 연산자일 때는 뺄셈 연산자가 되지만 단항 연산자로 사용되면 부호 변환 연산자가 되기도 합니다. 이 문제를 통해 추가로 알아 두기 바랍니다. 그리고 이 방법 말고 자주 사용되는 방법인 -1을 곱하는 방법을 사용해도 상관없습니다.

#include <stdio.h>

void main()

{

int nData = 0;

scanf("%d", &nData);

printf("%d\n", -nData);

}

**6.** 나머지 연산자의 응용 활용에 대한 문제입니다. 실제로 사용할 가능성도 매우 높은 예제이므로 원리를 잘 이해해 두기 바랍니다.

#include <stdio.h>

void main()

{

int nData = 0;

scanf("%d", &nData);

printf("%d:%d:%d\n",

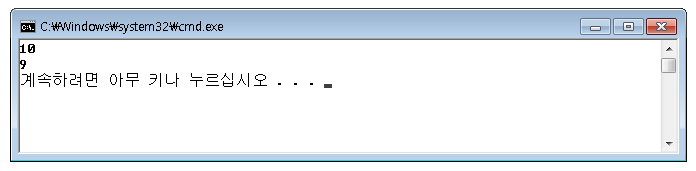
nData / 3600,

nData / 60 % 60,

nData % 60);

}

**7.** 단항 증감 연산자가 후위식으로 사용될 경우, 연산자의 우선순위가 가장 나중으로 밀립니다. 따라서 값을 출력한 후 감소합니다.



**제6장**

**1.** 3항 연산자와 관계 연산자를 조합하여 사용하는 방법에 대한 문제입니다. 다만, 어떤 경우에 대해 특정 연산을 수행하는 것이 아니라 출력할 값을 선택하는 형태로 사용되었습니다.

#include <stdio.h>

void main()

{

int x = 0, y = 0;

scanf("%d %d", &x, &y);

printf("%d\n", x > 10 && y > 10 ? 1 : 0);

}

**2.** 논리식의 생략에 대한 문제입니다. AND 논리 연산자 부분을 하나로 묶고 왼쪽부터 조건식의 결과가 참이 나올 때까지 오른쪽 방향으로 조건식에 대한 연산을 진행하면 됩니다. 만일, 참이 나오면 더 이상 연산을 수행하지 않습니다.

#include <stdio.h>

void main()

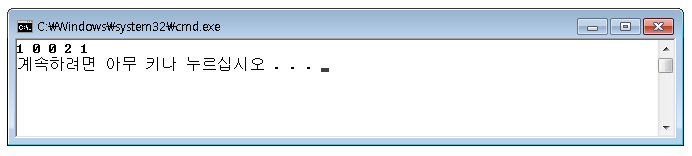
{

int nResult, w = 0, x = -1, y = -1, z = 1;

nResult = w++ || x++ && ++y || ++z;

printf("%d %d %d %d %d\n", w, x, y, z, nResult);

}



**3.** 3항 연산자를 사용하는 방법도 중요하지만, 이 문제에서는 최댓값을 구하는 기본 원리를 구현하는 것이 더 중요합니다. 가장 흔히 사용하는 방법으로는 최댓값이 저장될 변수를 사용하여 연산 과정에서 가장 큰 값이 그 변수에 항상 들어 있을 수 있도록 코드를 작성합니다.

#include <stdio.h>

void main()

{

int x = 0, y = 0, z = 0, nMax = 0;

scanf("%d %d %d", &x, &y, &z);

nMax = x > y ? x : y;

printf("%d\n", z > nMax ? nMax = z : nMax);

}

**제7장**

**1.** 임의의 값에서 최댓값을 구하는 문제입니다. 6장의 3번 문제와 원리가 같습니다. 다만 이 문제에서는 런타임에 사용자로부터 값이 순차적으로 입력된다는 정도의 차이만 있습니다. 이러한 경우에 최댓값이 저장될 변수를 따로 선언하는 것이 가장 효과적입니다.

#include <stdio.h>

void main()

{

int nInput = 0, nMax = 0;

scanf("%d", &nInput);

nMax = nInput;

scanf("%d", &nInput);

if(nInput > nMax) nMax = nInput;

scanf("%d", &nInput);

if(nInput > nMax) nMax = nInput;

scanf("%d", &nInput);

if(nInput > nMax) nMax = nInput;

scanf("%d", &nInput);

if(nInput > nMax) nMax = nInput;

printf("MAX : %d\n", nMax);

}

**2.** goto 문의 사용 방법 및 getch() 함수에 대한 문제입니다.

#include <conio.h>

void main()

{

char ch = 0;

BEGIN:

ch = getch();

if(ch == 'y' || ch == 'Y')

return;

else

goto BEGIN;

}

**3.** switch/case 문에 대한 문제이지만, if/else if 문을 사용해도 상관없습니다.

#include <stdio.h>

void main()

{

char ch = 0;

int x = 0, y = 0;

scanf("%c %d %d", &ch, &x, &y);

switch(ch)

{

case 'a':

printf("%d\n", x + y);

break;

case 'b':

printf("%d\n", x \* y);

break;

case 'c':

printf("%d\n", x - y);

break;

default:

puts("알 수 없는 명령.");

}

}

**제8장**

**1.** 반복문을 사용하는 방법을 확인해 보기 위한 가장 기본적인 문제라 할 수 있습니다.

#include <stdio.h>

void main()

{

int i = 0, nTotal = 0;

for(i = 1; i <= 100; ++i)

{

nTotal += i;

}

printf("%d\n", nTotal);

}

**2.** 반복문 및 나머지 연산자를 사용하는 방법에 대한 문제입니다.

#include <stdio.h>

void main()

{

int i = 0, nTotal = 0, nCount = 0;

for(i = 10; i <= 100; ++i)

{

if(i % 4 == 0)

{

nCount++;

nTotal += i;

printf("%d\n", i);

}

}

printf("Count: %d, Total: %d\n", nCount, nTotal);

}

**3.** 중첩된 반복문을 구성하는 방법에 대한 문제입니다.

<소스코드>

#include <stdio.h>

void main()

{

int i = 0, j = 0;;

for(i = 2; i < 10; ++i)

{

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

{

printf("%d \* %d = %d\n", i, j, i \* j);

}

putchar('\n');

}

}

**4.** 최댓값과 최솟값을 동시에 구하는 방법에 대한 문제인데, 최댓값이든 최솟값이든 원리는 사실상 같습니다.

#include <stdio.h>

void main()

{

int i = 0, nInput = 0, nMax = 0, nMin = 0;

scanf("%d", &nInput);

nMax = nMin = nInput;

for(i = 0; i < 4; ++i)

{

scanf("%d", &nInput);

if(nInput > nMax)

{

nMax = nInput;

continue;

}

if(nInput < nMin)

nMin = nInput;

}

printf("MAX: %d, MIN: %d\n", nMax, nMin);

}

**5.** 반복문을 이용한 문자열 처리 문제입니다. 이 문제를 제대로 이해하지 못할 경우, 문자 혹은 문자열 처리에 대한 내용을 이해하기 어려울 수 있습니다.

#include <stdio.h>

void main()

{

char ch = 0;

int nCount = 0;

while( (ch = getchar()) != '/')

{

if(ch == 't')

nCount++;

}

printf("%d\n", nCount);

}

**6.** 5번 문제와 목적이나 의미가 같습니다만, 좀더 정교하고 난이도가 높은 문제입니다. 또한 결과가 같다고 하더라도 과정이 전혀 다른 코드가 많이 있을 수 있습니다. 그러므로 다음의 코드가 정답이라고 생각하기 보다는 이와 같은 방법도 있다는 정도로 참고하기 바랍니다.

#include <stdio.h>

void main()

{

char ch = 0;

int nCount = 0, nFlag = 0;

while( (ch = getchar()) != '\n')

{

if(ch == ' ' || ch == '\t')

{

if(nFlag == 0)

{

nCount++;

nFlag = 1;

}

}

else

{

nFlag = 0;

}

}

if(nFlag == 0) nCount++;

printf("%d\n", nCount);

}

**7.** 이 문제에 속한 하위 네 문제는 모두 '프로그래밍의 기초'에 해당하는 문제입니다. 일종의 프로그래밍 퍼즐이라고 생각해도 좋습니다. 그리고 이러한 문제를 해결하기 위해 노력하는 것은 C언어의 실력을 향상시키는 것이라기보다 가장 근간이라 할 수 있는 프로그래밍 실력을 향상시키는 문제라고 할 수 있습니다. 그러므로 이 네 문제를 모두 완벽히 이해할 수 있어야 합니다. 혹시 문제를 풀지 않고 지금 이 글을 보고 있는 것이라면 즉시 책을 덮고 다시 도전해 보기 바랍니다. 개인적으로 이와 유사한 문제를 처음 접했을 때 약 2~3시간 정도가 걸렸던 것으로 기억합니다. 그러니 여러분도 최소 2~3 시간은 고민해 보기 바랍니다. 그리고 결과를 얻은 사람은 제가 작성한 답안과 비교해 보기 바랍니다. 혹시 방법이 다르다면 구체적으로 무엇이 다른지 꼭 확인해 보기 바랍니다.

**1)**

#include <stdio.h>

void main()

{

int i = 0, j = 0, nCount = 0;

for(i = 0; i < 5; ++i)

{

for(j = 0; j < 5; ++j)

{

printf("%d\t", ++nCount);

}

putchar('\n');

}

}

**2)**

#include <stdio.h>

void main()

{

int i = 0, j = 0, nCount = 0;

for(i = 0; i < 5; ++i)

{

for(j = 0; j < i + 1; ++j)

{

printf("%d\t", ++nCount);

}

putchar('\n');

}

}

**3)**

#include <stdio.h>

void main()

{

int i = 0, j = 0, nCount = 0;

for(i = 0; i < 5; ++i)

{

for(j = 0; j < 5; ++j)

{

if(i + j >= 4)

printf("%d\t", ++nCount);

else

putchar('\t');

}

putchar('\n');

}

}

**4)**

#include <stdio.h>

void main()

{

int i = 0, j = 0, nCount = 0;

for(i = 0; i < 5; ++i)

{

for(j = 0; j < 5 + i; ++j)

{

if(i + j >= 4)

printf("%d\t", ++nCount);

else

putchar('\t');

}

putchar('\n');

}

}

**제9장**

**1.** 이 문제는 memcmp() 함수의 사용 방법에 대한 문제입니다. 물론 다른 함수를 사용해서 같은 결과를 얻을 수 있지만 가장 효과적인 방법은 memcmp() 함수를 사용하는 것입니다.

#include <stdio.h>

#include <string.h>

void main()

{

char szName1[32] = {0}, szName2[32] = {0};

printf("Input name: ");

gets(szName1);

printf("Input name: ");

gets(szName2);

if(szName1[0] != 0 && memcmp(szName1, szName2, 2) == 0)

puts("같음");

else

puts("다름");

}

**2.** 1번 문제와 동일한 목적의 문제입니다. 두 문제를 통해서 int 형이든 char 형이든 실제로는 모두 메모리라는 사실을 인식할 수 있어야 합니다.

#include <stdio.h>

#include <string.h>

void main()

{

int x = 0, y = 0;

scanf("%d %d", &x, &y);

if(memcmp(&x, &y, sizeof(int)) == 0)

puts("같음");

else

puts("다름");

}

**3.** 메모리를 동적 할당하고 0으로 초기화 하는 가장 기본적인 메모리 관리 방법을 알고 있는지 확인하는 문제입니다.

#include <stdio.h>

#include <string.h>

#include <malloc.h>

void main()

{

void \*pMemory = NULL;

int nLength = 0;

scanf("%d", &nLength);

if(nLength < 10) nLength = 10;

if(nLength > 100) nLength = 100;

pMemory = malloc(nLength);

memset(pMemory, 0, nLength);

free(pMemory);

}

**4.** 메모리의 동적 할당 및 문자열 처리 코드를 직접 작성해 보기 위한 문제입니다. 참고로 문자열의 길이는 strlen() 함수를 이용하여 알아 낼 수 있습니다.

#include <stdio.h>

#include <string.h>

#include <malloc.h>

void main()

{

int nStrLength = 0, nMaxLength = 0;

char \*pszData = NULL;

scanf("%d", &nMaxLength);

pszData = (char\*)malloc(nMaxLength);

fflush(stdin);

gets(pszData);

while(\*(pszData + nStrLength) != 0)

{

nStrLength++;

}

printf("Length: %d\n", nStrLength);

free(pszData);

}

**5.** 문자열을 뒤집어 출력하기 위해서 출력할 문자열이 저장된 메모리의 주소에 문자열의 길이를 더한 위치에서 1씩 감소하여 문자를 출력하면 전체를 뒤집어 출력할 수 있습니다. 난이도가 높은 문제이며, 반드시 원리를 명확히 이해 해두어야 하는 중요한 문제입니다.

#include <stdio.h>

#include <string.h>

#include <malloc.h>

void main()

{

int nStrLength = 0, nMaxLength = 0;

char \*pszData = NULL;

scanf("%d", &nMaxLength);

pszData = (char\*)malloc(nMaxLength);

fflush(stdin);

gets(pszData);

while(\*(pszData + nStrLength) != 0)

{

nStrLength++;

}

for(nStrLength -= 1; nStrLength >= 0; --nStrLength)

{

putchar(\*(pszData + nStrLength));

}

putchar('\n');

free(pszData);

}

**6.** 이 문제는 2중 포인터와 메모리의 동적 관리를 명확히 이해하지 못했다면 해결할 수 없는 문제이며, 지금까지 살펴본 모든 문제 중에서 가장 난이도가 높은 문제입니다. 개인적으로 강의를 할 때는 이 문제를 과제로 제출 받기도 하며, 반드시 시험 문제로도 출제합니다. 이 문제 하나를 제대로 이해할 수 있다면 메모리와 포인터에 대해서 어느 정도 만족할만한 실력을 갖추었다고 생각해도 좋습니다.

#include <stdio.h>

#include <string.h>

#include <malloc.h>

void main()

{

int i = 0, nLineCount = 0, nMaxLength = 0;

char \*\*ppszData = NULL;

scanf("%d %d", &nLineCount, &nMaxLength);

ppszData = (char\*\*)malloc(sizeof(char\*) \* nLineCount);

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

{

ppszData[i] = (char\*)malloc(nMaxLength);

fflush(stdin);

gets(ppszData[i]);

}

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

{

printf("%p %s\n", ppszData[i], ppszData[i]);

}

free(ppszData);

}

**제10장**

**1.** 배열의 기본 문법에 대한 문제입니다.

#include <stdio.h>

void main()

{

int i = 0, anData[5] = {0};

for(i = 0; i < 5; ++i)

{

printf("%p\n", &anData[i]);

}

}

**2.** 배열의 기본 문법 및 배열의 이름이 갖는 의미가 무엇인지 확인할 목적의 문제입니다.

#include <stdio.h>

void main()

{

int i = 0, anData[5] = {0};

for(i = 0; i < 5; ++i)

{

printf("%p\n", anData + i);

}

}

**3.** 포인터 배열에 대한 이해 및 선택 정렬 알고리즘을 구현하는 방법을 알고 있는지 확인하는 문제입니다. 난이도가 높은 편은 아니지만 그렇다고 결코 가볍게 생각할 문제는 아니라 할 수 있습니다.

#include <stdio.h>

void main()

{

char \*pszTmp = NULL;

char \*apszData[5] = {

"유재석", "정형돈",

"박명수", "노홍철", "하하"};

int i = 0, j = 0;

for(i = 0; i < 4; ++i)

{

for(j = i + i; j < 5; ++j)

{

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

{

pszTmp = apszData[i];

apszData[i] = apszData[j];

apszData[j] = pszTmp;

}

}

}

for(i = 0; i < 5; ++i)

{

puts(apszData[i]);

}

}

**4.** 일명 '달팽이 배열' 문제입니다. 배열에 대한 이해보다는 프로그래밍 실력을 향상 시킬 목적의 문제라 할 수 있습니다. 참고로 이 문제에 대한 저의 답안은 인터넷에 알려진 몇몇 방법 중에서 가장 간결한 모습의 코드에 가까운 것입니다.

#include <stdio.h>

void main()

{

int arnData[5][5] = {0};

int i = 0, j = -1, k = 0;

int nLength = 5, nCounter = 0, nOffset = 1;

while(nCounter < 25)

{

for(k = 0; k < (nLength \* 2 -1); ++k)

{

if(k < nLength) j += nOffset;

else i += nOffset;

arnData[i][j] = ++nCounter;

}

nOffset \*= (-1);

nLength--;

}

for(i = 0; i < 5; ++i)

{

for(j = 0; j < 5; ++j)

{

printf("%d\t", arnData[i][j]);

}

putchar('\n');

}

}

**5.** 역방향 달팽이 배열문제입니다. 4번 문제의 답을 일부 수정하여 구현할 수 있는데 두 곳만 수정하면 됩니다.

#include <stdio.h>

void main()

{

int arnData[5][5] = {0};

int i = 0, j = 5, k = 0;

int nLength = 5, nCounter = 0, nOffset = 1;

while(nCounter < 25)

{

for(k = 0; k < (nLength \* 2 -1); ++k)

{

if(k < nLength) j -= nOffset;

else i += nOffset;

arnData[i][j] = ++nCounter;

}

nOffset \*= (-1);

nLength--;

}

for(i = 0; i < 5; ++i)

{

for(j = 0; j < 5; ++j)

{

printf("%d\t", arnData[i][j]);

}

putchar('\n');

}

}

**6.** 난이도가 높은 문제는 아니지만 매우 실질적인 문제라 할 수 있습니다.

#include <stdio.h>

void main()

{

int i = 0, j = 0;

int arnData[4][5] = {

{1, 2, 3, 4, 0},

{5, 6, 7, 8, 0},

{9, 10, 11, 12, 0},

{0, 0, 0, 0, 0}

};

for(i = 0; i < 3; ++i)

{

for(j = 0; j < 4; ++j)

{

arnData[i][4] += arnData[i][j];

}

}

for(i = 0; i < 5; ++i)

{

for(j = 0; j < 3; ++j)

{

arnData[3][i] += arnData[j][i];

}

}

for(i = 0; i < 4; ++i)

{

for(j = 0; j < 5; ++j)

{

printf("%d\t", arnData[i][j]);

}

putchar('\n');

}

}

**제11장**

**1.** 이미 앞서 이와 유사한 문제가 있었습니다. 이번에는 그 코드를 분리하여 별도의 함수로 만드는 방법을 알고 있는지 확인하는 기본적인 문제입니다.

#include <stdio.h>

int GetLength(char \*pszParam)

{

int nLength = 0;

while( \*(pszParam + nLength) != NULL)

{

nLength++;

}

return nLength;

}

void main()

{

char \*pszData = "Test string";

printf("[%d] %s\n", GetLength(pszData), pszData);

}

**2.** 1번 문제와 동일한 목적의 문제입니다.

#include <stdio.h>

#include <string.h>

void ReversePrint(char \*pszParam)

{

int nLength = 0;

nLength = strlen(pszParam) - 1;

for( ; nLength >= 0; --nLength)

{

putchar(\*(pszParam + nLength));

}

}

void main()

{

char \*pszData = "Test string";

ReversePrint(pszData);

putchar('\n');

}

**3.** 정렬 알고리즘을 하나의 함수로 구현할 수 있는지 확인하기 위한 문제입니다.

#include <stdio.h>

#include <string.h>

void SortData(int nCount, int anData[])

{

int nTmp = 0, i = 0, j = 0;

for(i = 0; i < nCount - 1; ++i)

{

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

{

if(anData[j] < anData[i])

{

nTmp = anData[j];

anData[j] = anData[i];

anData[i] = nTmp;

}

}

}

}

void main()

{

int i = 0, anData[5] = {4, 1, 3, 2, 5};

SortData(5, anData);

for(i = 0; i < 5; ++i)

{

printf("%d\t", anData[i]);

}

putchar('\n');

}

**4.** 구구단 출력과 관련한 코드를 함수로 만들 수 있는지 확인하기 위한 문제입니다.

#include <stdio.h>

#include <string.h>

void PrintMultiplication(int nParam)

{

int i = 0;

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

{

printf("%d \* %d = %d\n", nParam, i, nParam \* i);

}

putchar('\n');

}

void main()

{

PrintMultiplication(7);

PrintMultiplication(9);

}

**5.** 이 문제는 문자열이 저장된 메모리의 주소와 옵셋으로 인덱스를 계산해낼 수 있는 코드를 함수로 제작할 수 있는지 확인하는 문제입니다. 비교적 난이도가 높은 문제이며, 문자열과 배열에 대한 이론을 제대로 이해할 수 있도록 도와주는 문제라 할 수 있습니다. 따라서 반드시 이 문제에 대해 명확히 이해해야 합니다.

#include <stdio.h>

#include <string.h>

int FindChar(char \*pszParam, char ch)

{

int nIndex = 0;

for( ; \*(pszParam + nIndex) != 0; ++nIndex)

{

if(pszParam[nIndex] == ch)

return nIndex;

}

return -1;

}

void main()

{

char \*pszData = "Test string!";

printf("[%d], %s\n", FindChar(pszData, 't'), pszData);

printf("[%d], %s\n", FindChar(pszData, 'a'), pszData);

}

**6.** 다차원 배열에 대한 포인터 선언 문법 및 배열에 대한 문제입니다.

#include <stdio.h>

#include <string.h>

int GetTotal(int nCount, int (\*panData)[5])

{

int i = 0, j = 0, nTotal = 0;

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

{

for(j = 0; j < 5; ++j)

{

nTotal += panData[i][j];

}

}

return nTotal;

}

void main()

{

int anData[2][5] = {

{1, 2, 3, 4, 5},

{6, 7, 8, 9, 10}

};

printf("%d\n", GetTotal(2, anData));

}

**제12장**

**1.** 기본적인 문자 처리 함수 및 문자열 자체에 대해 알고 있는지 확인하기 위한 문제입니다.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void main()

{

char szBuffer[64] = {0};

int i = 0, nLength = 0;

gets(szBuffer);

nLength = strlen(szBuffer);

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

{

if(isupper(szBuffer[i]))

szBuffer[i] = tolower(szBuffer[i]);

else

szBuffer[i] = toupper(szBuffer[i]);

}

puts(szBuffer);

}

**2.** strlen() 함수의 사용 방법에 대한 문제입니다.

<소스코드>

#include <stdio.h>

#include <string.h>

void main()

{

char szBuffer1[64] = {0}, szBuffer2[64] = {0};

int nLength = 0;

gets(szBuffer1);

nLength = strlen(szBuffer1);

gets(szBuffer2);

if(strlen(szBuffer2) > nLength)

puts(szBuffer2);

else

puts(szBuffer1);

}

**3.** sprint() 함수 및 문자열이 저장된 메모리의 구조를 제대로 이해하고 있는지 확인하는 문제입니다. 기능은 단순하지만 결코 쉬운 문제는 아닙니다.

#include <stdio.h>

#include <string.h>

void main()

{

char szBufferDst[128] = {0}, szBuffer[64] = {0};

int nLength = 0;

gets(szBufferDst);

nLength = strlen(szBufferDst);

gets(szBuffer);

sprintf(szBufferDst + nLength, "%s", szBuffer);

puts(szBufferDst);

}

**4.** strcat() 함수의 활용에 대한 문제입니다. 결과적으로 3번 문제와 같습니다.

#include <stdio.h>

#include <string.h>

void main()

{

char szBufferDst[128] = {0}, szBuffer[64] = {0};

int nLength = 0;

gets(szBufferDst);

nLength = strlen(szBufferDst);

gets(szBuffer);

strcat(szBufferDst, szBuffer);

puts(szBufferDst);

}

**5.** 이 문제는 3번 문제와 유사하지만 별도의 메모리를 동적 할당하여 합친다는 점이 다릅니다. 그러나 3번 문제를 해결한 방법이 제가 만든 것과 달랐다면, 이 문제는 매우 난이도가 높은 문제가 될 수 있습니다. 그러므로 이 문제에 대해서 명확히 이해해야 합니다.

#include <stdio.h>

#include <string.h>

void main()

{

char \*pszResult = NULL;

char szBuffer1[64] = {0}, szBuffer2[64] = {0};

int nLength1 = 0, nLength2 = 0;

gets(szBuffer1);

gets(szBuffer2);

nLength1 = strlen(szBuffer1);

nLength2 = strlen(szBuffer2);

pszResult = (char\*)malloc(nLength1 + nLength2 + 1);

strcpy(pszResult, szBuffer1);

strcpy(pszResult + nLength1, szBuffer2);

puts(pszResult);

free(pszResult);

}

**6.** stricmp() 함수를 소개할 목적의 문제입니다. 만일 이 함수를 사용하지 않고 구현한다면, 두 문자열을 모두 대문자 혹은 소문자로 변환한 후 비교하면 됩니다.

#include <stdio.h>

#include <string.h>

void main()

{

char szBuffer1[64] = {0}, szBuffer2[64] = {0};

gets(szBuffer1);

gets(szBuffer2);

if(stricmp(szBuffer1, szBuffer2) == 0)

puts("Same");

else

puts("Different");

}

**7.** strtok() 함수의 사용 방법을 알고 있는지 확인하기 위한 문제입니다.

#include <stdio.h>

#include <string.h>

void main()

{

char szBuffer[128] = {0};

char \*pszSep = " \t\n";

char \*pszToken = NULL;

gets(szBuffer);

pszToken = strtok(szBuffer, pszSep);

while(pszToken != NULL)

{

puts(pszToken);

pszToken = strtok(NULL, pszSep);

}

}

**8.** 유니코드 문자열의 기본적인 사용 방법을 이해하고 있는지 확인하기 위한 문제입니다.

<소스코드>

#include <stdio.h>

#include <stdlib.h>

void main()

{

char szBuffer[128] = {0};

wchar\_t wszBuffer[128] = {0};

gets(szBuffer);

mbstowcs(wszBuffer, szBuffer, sizeof(szBuffer));

wprintf(L"%s\n", wszBuffer);

}

**제13장**

**1.** 그 동안 배운 것들을 좀더 실질적인 코드로 활용해 볼 목적으로 만들어진 문제이며 난이도가비교적 높은 프로그래밍 문제입니다. 논리적 구조가 아무리 간단해 보인다 해도, 그것을 코드로 옮기는 것은 전혀 다른 문제입니다. 그리고 이 문제에 대한 답안은 비교적 짧은 편에 속합니다. 이유는 '가위/바위/보'를 각각 '0/1/2'라고 했을 때 1 큰 값이 이기는 값이라는 원리(32~33행 참고)를 사용했기 때문입니다. 자세한 내용은 코드를 잘 살펴보시기 바랍니다. 그리고 이와 같은 원리를 사용하지 않고 경우의 수를 따져서 반영했다면 다수의 if문이 사용될 것입니다. 어떤 방법을 사용했든 결과가 바르다면 맞는 것으로 판단할 수 있습니다.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void PrintData(char\* pszData, int nParam)

{

if(nParam == 0) printf("%s: %s\n", pszData, "가위");

else if(nParam == 1) printf("%s: %s\n", pszData, "바위");

else if(nParam == 2) printf("%s: %s\n", pszData, "보");

}

void main()

{

int nUserInput = 0, nComputer = 0;

scanf("%d", &nUserInput);

if(nUserInput < 0) nUserInput = 0;

else if(nUserInput > 2) nUserInput %= 3;

srand( time(NULL) );

nComputer = rand() % 3;

PrintData("사용자", nUserInput);

PrintData("컴퓨터", nComputer);

if(nUserInput == nComputer)

{

puts("무승부");

}

else

{

nUserInput++;

nUserInput %= 3;

if(nUserInput == nComputer) puts("컴퓨터 승");

else puts("사용자 승");

}

}

**2.** 날짜를 계산하는 일은 매우 자주 발생하는 일이므로 이 문제를 잘 알아 두기 바랍니다.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void PrintDay(time\_t t)

{

struct tm \*ptime = {0};

ptime = localtime(&t);

printf("%04d-%02d-%02d\n",

ptime->tm\_year + 1900,

ptime->tm\_mon + 1, ptime->tm\_mday);

}

void main()

{

time\_t tCurrent = 0;

tCurrent = time(NULL);

PrintDay(tCurrent);

PrintDay(tCurrent + 60 \* 60 \* 24 \* 10);

PrintDay(tCurrent + 60 \* 60 \* 24 \* 100);

}

**3.** system() 함수에 대한 문제입니다.

<소스코드>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

void main()

{

printf("Hello, World!");

getch();

system("cls");

}

**제14장**

**1.** 자료형 정의에 대한 문법 문제입니다.

#include <stdio.h>

typedef unsigned char BYTE;

void main()

{

BYTE byData = 0;

}

**2.** 전역 변수에 대한 문법 문제입니다.

#include <stdio.h>

char g\_szBuffer[128] = {0};

void PrintData()

{

puts(g\_szBuffer);

}

void main()

{

gets(g\_szBuffer);

PrintData();

}

**3.** 외부 변수 선언에 대한 문법 문제입니다. 여러 소스 파일이 존재하는 경우를 실습해 볼 목적으로 만든 문제입니다.

#include <stdio.h>

extern char g\_szBuffer[128];

void PrintData();

void main()

{

gets(g\_szBuffer);

PrintData();

}

**4.** 정적 변수를 활용하기 위한 문법 문제입니다. 그러나 난이도가 좀 있는 문제라 할 수 있습니다. 많은 C언어 입문자들이 정적 변수에 대해 제대로 이해하지 못하거나 활용을 꺼리는 경향이 있습니다. 이 문제를 통해 그런 경향이 조금이라도 줄어 들기를 바랍니다.

#include <stdio.h>

#include <string.h>

void PrintString(char \*pszData)

{

static int nLength = 0;

int nCurLen = 0;

nCurLen = strlen(pszData);

if(nCurLen > nLength)

{

puts(pszData);

nLength = nCurLen;

}

}

void main()

{

PrintString("Test");

PrintString("String");

PrintString("Data");

PrintString("New");

}

**제15장**

**1.** 재귀 호출 함수에 대한 문법 및 활용 문제입니다. 코드가 어려운 것은 아니지만 이 문제를 시험에 출제하면 많은 사람이 틀리는 문제입니다. 수학적 이론을 프로그램으로 구현하는 것은 어려운 일일 수 있으나 그 만큼 중요한 문제이기도 합니다.

#include <stdio.h>

#include <string.h>

void Fibo(int nParam1, int nParam2)

{

if(nParam2 >= 60)

return;

printf("%d\t", nParam2);

Fibo(nParam2, nParam1 + nParam2);

}

void main()

{

Fibo(0, 1);

}

**2.** 재귀 호출을 활용하기 위한 실습 목적의 문제입니다.

#include <stdio.h>

#include <string.h>

void PutString(char\* pszData)

{

if(\*pszData == 0) return;

PutString(pszData + 1);

putchar(\*pszData);

}

void main()

{

PutString("TestData!");

putchar('\n');

}

**3.** 함수 포인터 변수에 대한 문법 문제입니다.

#include <stdio.h>

#include <string.h>

char\* TestFunc(int nParam1, int nParam2)

{

return NULL;

}

void main()

{

char\* (\*pfTestFunc)(int, int) = TestFunc;

}

**4.** 함수 포인터 변수에 대한 문법 문제입니다. 그러나 3번 문제보다는 난이도가 높습니다.

#include <stdio.h>

#include <string.h>

char\* TestFunc(int nParam1, int nParam2)

{

return NULL;

}

int ResultFunc( char\* (\*pfTestFunc)(int, int))

{

puts("ResultFunc() - call");

return 0;

}

void main()

{

int (\*pfResultFunc)(char\* (\*)(int, int)) = ResultFunc;

pfResultFunc(TestFunc);

}

**5.** 호출 규약을 포함한 함수 포인터 변수에 대한 문법 문제입니다.

#include <stdio.h>

#include <string.h>

int \_\_stdcall TestFunc()

{

puts("TestFunc() - call");

return 0;

}

void main()

{

int (\_\_stdcall \*pfTestFunc)() = TestFunc;

pfTestFunc();

}

**제16장**

**1.** 기본적인 구조체 문법에 대한 문제입니다.

#include <stdio.h>

#include <string.h>

typedef struct \_STUDENT

{

char szName[32];

char szNumber[12];

char szPhoneNumber[16];

int anScore[3];

int nRank;

} STUDENT;

void main()

{

STUDENT data = {0};

}

**2.** 구조체의 초기화 및 멤버 접근 문법에 대한 문제입니다.

#include <stdio.h>

#include <string.h>

typedef struct \_STUDENT

{

char szName[32];

char szNumber[12];

char szPhoneNumber[16];

int anScore[3];

int nRank;

} STUDENT;

void main()

{

STUDENT data = {

"홍길동", "1234567",

"010-1234-5678",

{95, 80, 85},

10

};

printf("%s %s %s\n%d %d %d\n%d\n",

data.szName,

data.szNumber,

data.szPhoneNumber,

data.anScore[0], data.anScore[1], data.anScore[2],

data.nRank);

}

**제17장**

**1.** 버블 정렬에 대한 알고리즘 구현 문제입니다.

<소스코드>

#include <stdio.h>

#include <malloc.h>

void main()

{

int \*pnData = NULL;

int i = 0, j = 0, nTmp = 0;

pnData = (int\*)malloc(sizeof(int) \* 5);

for(i = 0; i < 5; ++i)

{

scanf("%d", pnData + i);

}

for(i = 4; i >= 1; --i)

{

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

{

if(pnData[j] > pnData[j + 1])

{

nTmp = pnData[j];

pnData[j] = pnData[j + 1];

pnData[j + 1] = nTmp;

}

}

}

for(i = 0; i < 5; ++i)

printf("%d\t", pnData[i]);

putchar('\n');

free(pnData);

}

**2.** 간단한 스택을 구현할 수 있는지 확인하기 위한 문제입니다. 코드가 어려운 것은 아니지만 매우 중요한 문제입니다.

#include <stdio.h>

int g\_anStack[5] = {0};

int g\_nStackpointer = -1;

int Push(int nData)

{

if(g\_nStackpointer >= 4 || nData < 0)

return 0;

g\_nStackpointer++;

g\_anStack[g\_nStackpointer] = nData;

return 1;

}

int Pop(int \*pnData)

{

if(g\_nStackpointer < 0)

return 0;

\*pnData = g\_anStack[g\_nStackpointer];

g\_nStackpointer--;

return 1;

}

void main()

{

int i = 0, nData = 0;

for(i = 1; i <= 5; ++i)

Push(i);

for(i = 1; i <= 5; ++i)

{

if(Pop(&nData))

{

printf("%d\n", nData);

}

}

}

**3.** 콜백 함수 및 qsort() 함수를 실습해 볼 목적으로 만든 문제입니다.

#include <stdio.h>

#include <stdlib.h>

int CallBackCompare( const void \*arg1, const void \*arg2 )

{

return \*((int\*)arg2) - \*((int\*)arg1);

}

void main()

{

int i = 0, anData[5] = {4, 1, 5, 3, 2};

qsort(anData, 5, sizeof(int), CallBackCompare);

for(i = 0; i < 5; ++i)

printf("%d\t", anData[i]);

putchar('\n');

}

**제18장**

**1.** stdin, stdout 파일의 의미를 다시 한번 명확히 확인하기 위한 문제입니다.

#include <stdio.h>

void main()

{

char szBuffer[128] = {0};

fgets(szBuffer, sizeof(szBuffer), stdin);

fputs(szBuffer, stdout);

}

**2.** 텍스트 모드 파일의 기본적인 활용 방법을 알고 있는지 확인하기 위한 문제입니다.

#include <stdio.h>

void main()

{

FILE \*fp = NULL;

char szBuffer[128] = {0};

fp = fopen("TestFile.txt", "a+");

fgets(szBuffer, sizeof(szBuffer), stdin);

fputs(szBuffer, fp);

fclose(fp);

}

**3.** 파일의 복사를 구현하는 방법을 좀더 쉽게 경험해 볼 수 있도록 한 문제입니다. 이 문제를 해결할 수 없다면 4번 문제도 해결할 수 없습니다. 어떤 의미에서 이 문제는 4번 문제를 해결과 관련된 지식과 코딩 능력을 향상시키기 위한 문제입니다.

<소스코드>

#include <stdio.h>

void main()

{

FILE \*fpSrc = NULL, \*fpDst = NULL;

char szBuffer[1024] = {0};

fpSrc = fopen("TestFile.txt", "r");

if(fpSrc == NULL) return;

fpDst = fopen("DstFile.txt", "w");

if(fpSrc == NULL)

{

fclose(fpSrc);

return;

}

while(fgets(szBuffer, sizeof(szBuffer), fpSrc))

{

fprintf(fpDst, "%s", szBuffer);

}

fclose(fpSrc);

fclose(fpDst);

}

**4.** 바이너리 모드 파일을 다룰 수 있는지 확인해 보기 위한 문제입니다. 잘 생각해보면 그리 어려운 문제가 아니지만 실제로 구현하려면 생각해야 할 것들이 꽤 많은 문제입니다. 이런 이유로 난이도가 비교적 높은 편에 속하며, 실제로 사용할 일도 많은 매우 중요한 문제입니다. 그리고 이 문제의 답에서 대상 파일에 대한 쓰기 시도가 실패했을 경우를 고려하지 않고 있습니다. 그러므로 혹시 문제를 제대로 해결하지 못해서 답을 작성하지 못했다면 쓰기 실패에 대한 예외처리 코드를 추가해 보는 실습부터 진행해 보기 바랍니다.

#include <stdio.h>

void main(int argc, char \*argv[])

{

FILE \*fpSrc = NULL, \*fpDst = NULL;

unsigned char byBuffer[4096] = {0};

long lTotal = 0, lRead = 0, lSrcSize = 0;

if(argc < 3) return;

fpSrc = fopen(argv[1], "rb");

if(fpSrc == NULL) return;

fseek(fpSrc, 0, SEEK\_END);

lSrcSize = ftell(fpSrc);

rewind(fpSrc);

fpDst = fopen(argv[2], "wb");

if(fpSrc == NULL)

{

fclose(fpSrc);

return;

}

while( (lRead = fread(byBuffer, 1, 4096, fpSrc)) > 0)

{

fwrite(byBuffer, 1, lRead, fpDst);

lTotal += lRead;

printf("%ld%%\n", lTotal \* 100 / lSrcSize);

}

fclose(fpSrc);

fclose(fpDst);

}

**제19장**

**1.** 매크로 정의와 관련한 문법 문제입니다.

#include <stdio.h>

#define LOWORD(data) (data & 0x0000FFFF)

void main(int argc, char \*argv[])

{

int nData = 0x12345678;

printf("%X, %X\n", nData, LOWORD(nData));

}

**2.** 조건부 컴파일에 대한 문법 문제입니다.

#include <stdio.h>

void main(int argc, char \*argv[])

{

#ifdef \_DEBUG

puts("Debug mode Hello, World!");

#else

puts("Release mode Hello, World!");

#endif

}

**제20장**

다음의 코드는 완료 프로젝트를 구현한 결과입니다. 노드의 위치와 상관 없이 동일한 삭제 코드가 적용되도록 하기 위해 더미 노드를 사용하는 구조로 작성된 것입니다. 그리고 사용자 입력이나 출력에 대한 인터페이스 부분과 실제로 리스트에 접근하는 부분의 코드가 대부분 분리되어 있습니다. 반드시 이와 같이 구현해야 하는 강제 사항은 아니지만 권장 사항이 되는 것이니 참고하시기 바랍니다. 만일 스스로 구현하는데 실패해서 먼저 답을 보는 것이라면, 이 코드를 철저히 분석하여 자신의 것으로 만들어 보기 바랍니다. 가끔은 그러는 편이 더 효율적일 수도 있습니다. 그러나 언젠가는 반드시 여러분 혼자 스스로 구현할 수 있어야 합니다. 혹시 이후로 C++를 공부할 예정이라면 이 정도의 예제는 외워서 전체를 코딩할 수 있을 정도로 연습하는 것이 좋습니다.

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

/////////////////////////////////////////////////////////////////////////

#define DATA\_FILE\_NAME "Address.dat"

/////////////////////////////////////////////////////////////////////////

void ReleaseList();

/////////////////////////////////////////////////////////////////////////

typedef struct \_USERDATA

{

char szName[32];

char szPhone[32];

struct \_USERDATA \*pNext;

} USERDATA;

USERDATA g\_Head = {0};

/////////////////////////////////////////////////////////////////////////

USERDATA\* FindNode(char \*pszName)

{

USERDATA \*pTmp = g\_Head.pNext;

while(pTmp != NULL)

{

if(strcmp(pTmp->szName, pszName) == 0)

return pTmp;

pTmp = pTmp->pNext;

}

return NULL;

}

/////////////////////////////////////////////////////////////////////////

int AddNewNode(char \*pszName, char \*pszPhone)

{

USERDATA \*pNewUser = NULL;

if(FindNode(pszName) != NULL) return 0;

pNewUser = (USERDATA\*)malloc(sizeof(USERDATA));

memset(pNewUser, 0, sizeof(USERDATA));

sprintf(pNewUser->szName, "%s", pszName);

sprintf(pNewUser->szPhone, "%s", pszPhone);

pNewUser->pNext = NULL;

pNewUser->pNext = g\_Head.pNext;

g\_Head.pNext = pNewUser;

return 1;

}

/////////////////////////////////////////////////////////////////////////

void Add()

{

char szName[32] = {0};

char szPhone[32] = {0};

printf("Input name : ");

fflush(stdin);

gets(szName);

printf("Input phone number : ");

fflush(stdin);

gets(szPhone);

AddNewNode(szName, szPhone);

// puts("Add");

// getch();

}

/////////////////////////////////////////////////////////////////////////

void Search()

{

char szName[32] = {0};

USERDATA \*pNode = NULL;

printf("Input name : ");

fflush(stdin);

gets(szName);

pNode = FindNode(szName);

if(pNode != NULL)

{

printf("[%p] %s\t%s [%p]\n",

pNode,

pNode->szName, pNode->szPhone,

pNode->pNext);

}

else

{

puts("ERROR: 데이터를 찾을 수 없습니다.");

}

getch();

}

/////////////////////////////////////////////////////////////////////////

void PrintAll()

{

USERDATA \*pTmp = g\_Head.pNext;

while(pTmp != NULL)

{

printf("[%p] %s\t%s [%p]\n",

pTmp,

pTmp->szName, pTmp->szPhone,

pTmp->pNext);

pTmp = pTmp->pNext;

}

getch();

}

/////////////////////////////////////////////////////////////////////////

int RemoveNode(char \*pszName)

{

USERDATA \*pPrevNode = &g\_Head;

USERDATA \*pDelete = NULL;

while(pPrevNode->pNext != NULL)

{

pDelete = pPrevNode->pNext;

if(strcmp(pDelete->szName, pszName) == 0)

{

pPrevNode->pNext = pDelete->pNext;

free(pDelete);

return 1;

}

pPrevNode = pPrevNode->pNext;

}

return 0;

}

/////////////////////////////////////////////////////////////////////////

void Remove()

{

char szName[32] = {0};

USERDATA \*pNode = NULL;

printf("Input name : ");

fflush(stdin);

gets(szName);

RemoveNode(szName);

}

/////////////////////////////////////////////////////////////////////////

//UI출력

int PrintUI()

{

int nInput = 0;

system("cls");

printf("[1]Add\t[2]Search\t[3]Print all\t[4]Remove\t[0]Exit\n:");

scanf("%d", &nInput);

return nInput;

}

/////////////////////////////////////////////////////////////////////////

int LoadList(char \*pszFileName)

{

FILE \*fp = NULL;

USERDATA user = {0};

fp = fopen(pszFileName, "rb");

if(fp == NULL)

{

// puts("ERROR: 리스트 파일을 읽기모드로 열지 못했습니다.");

// getch();

return 0;

}

ReleaseList();

while(fread(&user, sizeof(USERDATA), 1, fp))

{

AddNewNode(user.szName, user.szPhone);

}

fclose(fp);

return 0;

}

/////////////////////////////////////////////////////////////////////////

int SaveList(char \*pszFileName)

{

FILE \*fp = NULL;

USERDATA \*pTmp = g\_Head.pNext;

fp = fopen(pszFileName, "wb");

if(fp == NULL)

{

puts("ERROR: 리스트 파일을 쓰기모드로 열지 못했습니다.");

getch();

return 0;

}

while(pTmp != NULL)

{

if(fwrite(pTmp, sizeof(USERDATA), 1, fp) != 1)

{

printf("ERROR: %s에 대한 정보를 저장하는데 실패했습니다.\n", pTmp->szName);

}

pTmp = pTmp->pNext;

}

fclose(fp);

return 1;

}

/////////////////////////////////////////////////////////////////////////

void ReleaseList()

{

USERDATA \*pTmp = g\_Head.pNext;

USERDATA \*pDelete = NULL;

while(pTmp != NULL)

{

pDelete = pTmp;

pTmp = pTmp->pNext;

free(pDelete);

}

memset(&g\_Head, 0, sizeof(USERDATA));

}

/////////////////////////////////////////////////////////////////////////

void main()

{

int nMenu = 0;

LoadList(DATA\_FILE\_NAME);

while( (nMenu = PrintUI()) != 0 )

{

switch(nMenu)

{

case 1: //Add

Add();

break;

case 2: //Search

Search();

break;

case 3: //Print all

PrintAll();

break;

case 4: //Remove

Remove();

break;

}

}

SaveList(DATA\_FILE\_NAME);

ReleaseList();

}