

8-1. 포인터

♻ 포인터란?

메모리에 있는 data의 주소를 포인터라 한다. 즉, 포인터는 메모리에 있는 data의 위치를 표현한 수치이다. 메모리의 주소는 16진수 8자리로 구성되어 있으며 이 주소를 포인터라 한다.

♻ 포인터 변수란?

포인터 값(주소 값)을 보관하기 위해 만든 변수를 포인터 변수라 한다. 따라서 포인터 변수는 어떠한 data가 있는 곳의 주소값을 보관하며 이때 "포인터 변수가 data를 가리키고 있다." 라고 표현한다. data의 크기가 1byte 이상이라 여러 개의 주소가 있다고 해도 data의 시작주소가 대표주소가 되며 data의 포인터 값이라 하면 이 시작주소를 의미한다.

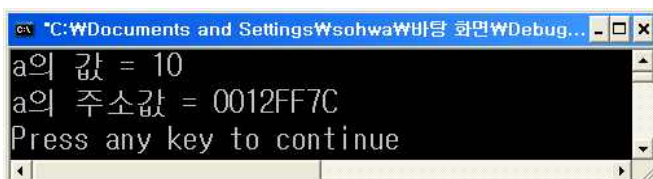
포인터 변수는 아래의 특징을 가지고 있다.

- ① 변수를 만들 때 앞쪽에 * 기호를 붙이면 포인터 변수가 만들어 진다.
- ② 포인터 변수는 오직 주소값만 보관 할 수 있다. 단, 자신이 만들어 질 때 사용한 자료형과 같은 자료형을 가지는 data의 주소값만 보관시킬 수 있다.
- ③ 포인터 변수의 크기는 자료형에 상관없이 4byte이다. (DOS 운영체제 에서는 2byte이다.)
- ④ 포인터 변수는 +1할 때마다 자신이 주소값을 보관 할 수 있는 data의 크기만큼 증가한다.
- ⑤ 포인터 변수를 사용하면서 앞쪽에 *를 붙이면 역참조가 된다. 역참조를 이용해 포인터가 가리키는 변수의 data를 변경 할 수 있다.
 - ※ 역참조란 포인터 변수가 가지고 있는 주소를 따라 이동해 그 안의 data를 가져오는 것이다. 즉, 포인터 변수가 가리키는 곳에 있는 값을 꺼내온다.

따라서 포인터 변수를 이용하면 변수명으로 호출이 불가능한 변수도 주소값을 이용해 호출하고 그 값을 변경 할 수 있다. 또한 메모리에 연달아 저장되어 있는 값을 순차적으로 호출할 경우에도 유용하게 사용할 수 있다.

예제 8-1 : 변수의 주소값을 구하는 방법

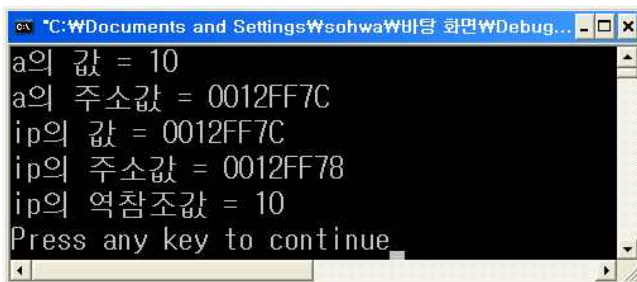
```
#include <stdio.h>
void main ( ) {
    int a = 10;
    printf("a의 값 = %d\n", a);
    printf("a의 주소값 = %p\n", &a);    // 변수앞에 &를 붙이면 변수의 주소값이 나온다.
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\Debug...
a의 값 = 10
a의 주소값 = 0012FF7C
Press any key to continue
```

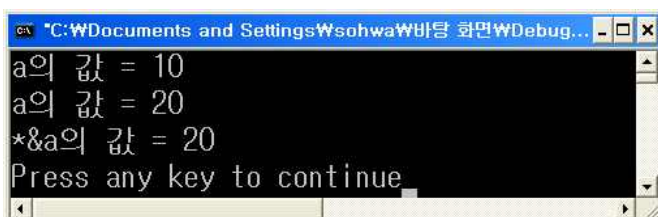
예제 8-2 : 포인터 변수의 사용방법 및 의미

```
#include <stdio.h>
void main ( ) {
    int a = 10;
    int *ip;           // 포인터 변수는 만들 때 변수명 앞에 *를 붙여 만든다.
                       // int* ip;, int * ip; 도 모두 같은 의미를 가진다.
    ip = &a;           // 포인터 변수에 주소값을 넣는 방법.
    printf("a의 값 = %d\n", a);
    printf("a의 주소값 = %p\n", &a);    // 변수앞에 &를 붙이면 변수의 주소값이 나온다.
    printf("ip의 값 = %p\n", ip);       // ip안에 있는 주소값이 나온다.
    printf("ip의 주소값 = %p\n", &ip);  // ip의 주소값이 나온다.
    printf("ip의 역참조값 = %d\n", *ip); // ip안에 있는 주소값을 따라가 그안에 있는
                                         // data인 10을 꺼나온다.
}
```



예제 8-3 : 역참조를 이용한 변수 값의 변경

```
#include <stdio.h>
void main ( ) {
    int a = 10;
    int *ip;           // 포인터 변수는 만들 때 변수명 앞에 *를 붙여 만든다.
    ip = &a;           // 포인터 변수에 주소값을 넣는 방법.
    printf("a의 값 = %d\n", a);
    *ip = 20;           // 역참조를 통해 변수 a의 값을 20으로 변경한다.
    printf("a의 값 = %d\n", a);
    printf("*a의 값 = %d\n", *a); // a의 주소값을 역참조 했음으로 a가 된다.
}
```

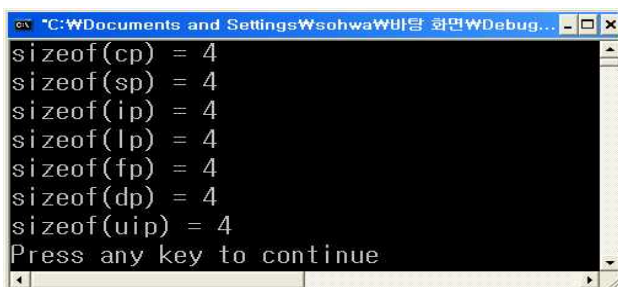


예제 8-4 : 포인터 변수는 오직 같은 자료형을 가지는 변수의 주소값 만을 보관할 수 있다.

```
#include <stdio.h>
void main ( ) {
    int a = 10;
    double b = 3.14;
    char c = 'A';
    int *ip;
    ip = 3;           // 포인터 변수에 주소가 아닌 값을 넣었음으로 에러
    ip = &b;          // int형 포인터변수에 double형 변수의 주소값을 넣었음으로 에러
    ip = &c;          // int형 포인터변수에 char형 변수의 주소값을 넣었음으로 에러
    ip = 0x0012FF7C;  // 주소값과 같은 형태이지만 주소값이 아님으로 에러
    ip = a;           // int형 포인터 변수에 int를 넣었음으로 에러
}
```

예제 8-5 : 모든 포인터 변수의 크기는 4byte인 것을 확인하는 예제

```
#include <stdio.h>
void main ( ) {
    char *cp;   short *sp;   int *ip;   long *lp;
    float *fp;  double *dp;  unsigned int *uip;
    printf("sizeof(cp) = %d\n", sizeof(cp) );
    printf("sizeof(sp) = %d\n", sizeof(sp) );
    printf("sizeof(ip) = %d\n", sizeof(ip) );
    printf("sizeof(lp) = %d\n", sizeof(lp) );
    printf("sizeof(fp) = %d\n", sizeof(fp) );
    printf("sizeof(dp) = %d\n", sizeof(dp) );
    printf("sizeof(uip) = %d\n", sizeof(uip) );
}
```



예제 8-6 : 포인터 변수가 +1 될 때마다 얼마씩 값이 증가하는지 확인하는 예제

```
#include <stdio.h>
void main ( ) {
    char a = 'Z';
    int b = 10;
    double c = 3.141592;
```

```

char *cp = &a;
int *ip = &b;
double *dp = &c;
printf("cp = %p\n", cp);           // a의 주소값이 출력된다.
printf("cp+1 = %p\n", cp+1);       // a가 char임으로 cp보다 1증가한 값이 나온다.
printf("cp+2 = %p\n", cp+2);       // a가 char임으로 cp보다 2증가한 값이 나온다.
printf("\n");
printf("ip = %p\n", ip);
printf("ip+1 = %p\n", ip+1);       // b가 int임으로 ip보다 4증가한 값이 나온다.
printf("ip+2 = %p\n", ip+2);       // b가 int임으로 ip보다 8증가한 값이 나온다.
printf("\n");
printf("dp = %p\n", dp);
printf("dp+1 = %p\n", dp+1);       // c가 double임으로 dp보다 8증가한 값이 나온다.
printf("dp+2 = %p\n", dp+2);       // c가 double임으로 dp보다 16증가한 값이 나온다.
}

```

```

C:\Documents and Settings\sohwa\바탕 화면\Debug...
cp = 0012FF7C
cp+1 = 0012FF7D
cp+2 = 0012FF7E

ip = 0012FF78
ip+1 = 0012FF7C
ip+2 = 0012FF80

dp = 0012FF70
dp+1 = 0012FF78
dp+2 = 0012FF80
Press any key to continue

```

중첩 포인터 변수란?

포인터 변수도 변수임으로 메모리에 만들어 지고 메모리에 만들어 지기 때문에 주소값이 있다. 이런 포인터 변수의 주소 값을 저장하는 포인터 변수를 중첩 포인터 변수라 한다. 또한 중첩포인터 변수의 주소값을 보관하는 포인터 변수를 삼중첩 포인터 변수라 한다. 중첩 포인터도 포인터 임으로 특성은 일반 포인터와 동일하다.

예제 8-7 : 중첩포인터 변수 만들기

```

#include <stdio.h>
void main ( ) {
    double a = 3.141592;
    double *dp = &a;           // dp는 double *임으로 double형 변수의 주소값을 가질 수 있다.
    double **dp2 = &dp;        // dp2는 double **임으로 double *형 변수의 주소 값을 가질
                                // 수 있다. double * * dp2;, double** dp2; double* *dp2;,
                                // double ** dp2; 모두 같은 의미이다.
}

```

```

printf(" a = %lf\n", a);      // a의 값이 출력된다.
printf("&a = %p\n", &a);      // a의 주소값이 출력된다.
printf("Wn");
printf(" dp = %p\n", dp);     // dp의 값(a의 주소값)이 출력된다.
printf("&dp = %p\n", &dp);     // dp의 주소값(메모리에서 dp의 위치)이 출력된다.
printf("*dp = %lf\n", *dp);   // dp가 가리키는 변수(변수 a)의 data값이 출력된다.
printf("Wn");
printf("  dp2 = %p\n", dp2);  // dp2의 값(dp의 주소값)이 출력된다.
printf(" &dp2 = %p\n", &dp2); // dp2의 주소값(메모리에서 dp2의 위치)이 출력된다.
printf(" *dp2 = %p\n", *dp2); // dp2가 가리키는 변수(변수 dp)의 data값이 출력된다.
printf("**dp2 = %lf\n", **dp2); // dp2가 가리키는 변수(변수 dp)가 가리키는 변수(a)
                                // 의 data값이 출력된다.
}

```

```

C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
a = 3.141592
&a = 0012FF78

dp = 0012FF74
&dp = 0012FF74
*dp = 3.141592

dp2 = 0012FF74
&dp2 = 0012FF70
*dp2 = 0012FF78
**dp2 = 3.141592
Press any key to continue

```

예제 8-8 : 삼중첩 포인터의 사용방법

```

#include <stdio.h>
void main ( ) {
    short a = 10;
    short *sp = &a;      // sp는 short *임으로 short형 변수 a의 주소값을 가질 수 있다.
    short **sp2 = &sp;   // sp2는 short **임으로 short *형 변수의 주소값을 가질 수 있다.
    short ***sp3 = &sp2; // sp3는 short ***임으로 short **형 변수의 주소값을
                        // 가질 수 있다.
    printf(" &sp2 = %p\n", &sp2 );
    printf("Wn");
    printf("  sp3 = %p\n", sp3); // sp3에 있는 sp2의 주소값이 나온다.
    printf(" &sp3 = %p\n", &sp3); // sp3의 주소값이 나온다.
    printf(" *sp3 = %p\n", *sp3); // sp3를 역참조한 sp2의 값이 나온다.
    printf(" **sp3 = %p\n", **sp3); // sp3를 두 번 역참조한 sp의 값이 나온다.
    printf(" ***sp3 = %d\n", ***sp3); // sp3를 세 번 역참조한 a의 값이 나온다.
}

```

```

C:\WDocuments and Settings\sohwa\바탕 화면\WDebug...
&sp2 = 0012FF74

    sp3 = 0012FF74
    &sp3 = 0012FF70
    *sp3 = 0012FF78
    **sp3 = 0012FF7C
    ***sp3 = 10
Press any key to continue

```

♻️ 배열과 포인터와의 관계

배열은 메모리에 연달아 여러 개의 변수를 만들 수 있게 해주는 도구이다. 사용자가 배열을 만들면 컴파일러는 메모리에 배열에 전체 크기에 해당하는 메모리를 할당받고 할당받은 메모리의 시작 주소값을 배열에 이름에 저장한다. 따라서 배열의 이름은 포인터 이며 상수이다. 또한 배열의 이름을 역참조하면 배열안에 있는 첫 번째 원소의 값이 나오게 된다. 종합하면 배열의 이름은 배열에 있는 첫 번째 원소의 주소값을 가지고 있는 포인터 상수가 된다.

- ① 1차원 배열의 경우에는 첫 번째 원소가 [0]번 원소 임으로 배열의 이름은 첫 번째 원소에 대한 포인터 상수가 된다.
- ② 2차원 배열의 경우에는 첫 번째 원소가 [0]번 행이 됨으로 배열의 이름은 한 행 전체를 가리키는 포인터 상수가 된다.
- ③ 3차원 배열의 경우에는 첫 번째 원소가 [0]번 면이 됨으로 배열의 이름은 한 면 전체를 가리키는 포인터 상수가 된다.

[] 연산자의 의미 : 배열안에 있는 원소를 호출하기 위해 사용하는 [] 연산자는 포인터[정수]의 형태로 사용되며 컴파일시 *(포인터+정수) 로 변경되어 실행된다. (배열을 만들 때 사용하는 []와는 의미가 다르다 배열을 만들때는 배열의 크기를 정하기 위해 []를 사용한다.)

※ 예1) arr[3] 은 컴파일 시 *(arr+3) 으로 변경되어 실행된다.

※ 예2) arr[2][3] 은 컴파일 시 *(arr[2]+3) 으로 변경되고 다시 *(*(arr+2) +3) 으로 변경되어 실행된다.

예제 8-9 : 배열의 이름이 가지는 의미

```

#include <stdio.h>
void main ( ) {
    short arr[3] = {10,20,30};
    printf(" arr = %p\n", arr);
    printf(" &arr[0] = %p\n", &arr[0]);
}

```

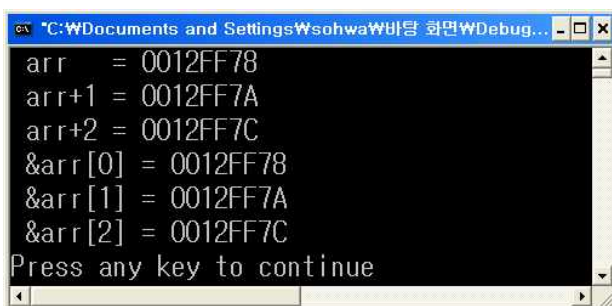
```

C:\WDocuments and Settings\sohwa\바탕 화면\WDebug...
arr = 0012FF78
&arr[0] = 0012FF78
Press any key to continue

```

예제 8-10 : 배열의 이름 +정수를 했을 때의 결과

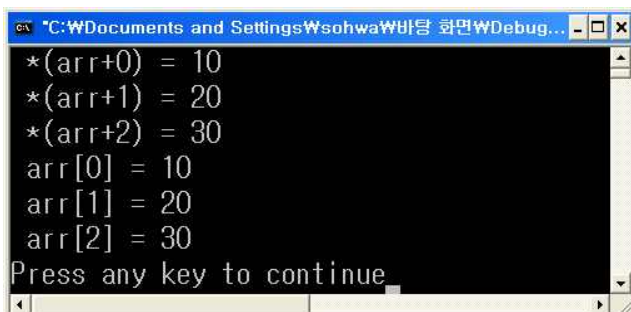
```
#include <stdio.h>
void main ( ) {
    short arr[3] = {10,20,30};
    printf(" arr   = %p\n", arr);
    printf(" arr+1 = %p\n", arr+1);
    printf(" arr+2 = %p\n", arr+2);
    printf(" &arr[0] = %p\n", &arr[0]);
    printf(" &arr[1] = %p\n", &arr[1]);
    printf(" &arr[2] = %p\n", &arr[2]);
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
arr   = 0012FF78
arr+1 = 0012FF7A
arr+2 = 0012FF7C
&arr[0] = 0012FF78
&arr[1] = 0012FF7A
&arr[2] = 0012FF7C
Press any key to continue
```

예제 8-11 : 배열의 이름을 역참조 했을 때의 결과

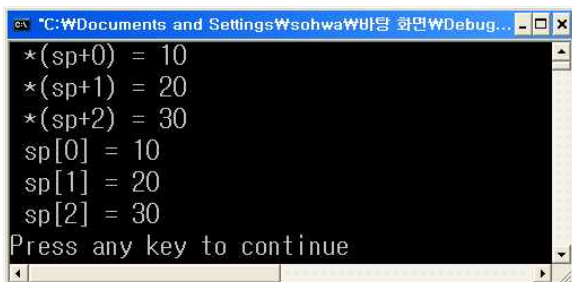
```
#include <stdio.h>
void main ( ) {
    short arr[3] = {10,20,30};
    printf(" *(arr+0) = %d\n", *(arr+0) );
    printf(" *(arr+1) = %d\n", *(arr+1) );
    printf(" *(arr+2) = %d\n", *(arr+2) );
    printf(" arr[0] = %d\n", arr[0]);
    printf(" arr[1] = %d\n", arr[1]);
    printf(" arr[2] = %d\n", arr[2]);
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
*(arr+0) = 10
*(arr+1) = 20
*(arr+2) = 30
arr[0] = 10
arr[1] = 20
arr[2] = 30
Press any key to continue
```


예제 8-12 : 포인터 변수를 이용해 배열안의 원소에 접근하는 방법

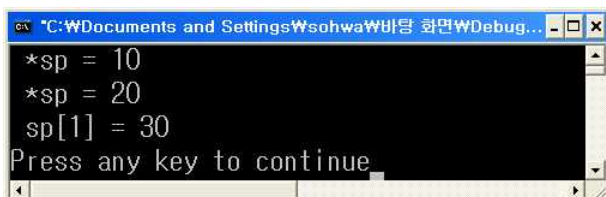
```
#include <stdio.h>
void main ( ) {
    short arr[3] = {10,20,30};
    short *sp;
    sp = arr;
    printf(" *(sp+0) = %d\n", *(sp+0) );
    printf(" *(sp+1) = %d\n", *(sp+1) );
    printf(" *(sp+2) = %d\n", *(sp+2) );
    printf(" sp[0] = %d\n", sp[0]);
    printf(" sp[1] = %d\n", sp[1]);
    printf(" sp[2] = %d\n", sp[2]);
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
*(sp+0) = 10
*(sp+1) = 20
*(sp+2) = 30
sp[0] = 10
sp[1] = 20
sp[2] = 30
Press any key to continue
```

예제 8-13 : 포인터 변수를 이용한 배열안의 원소에 대한 접근

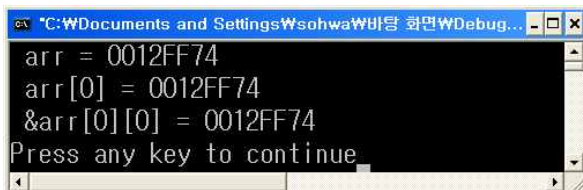
```
#include <stdio.h>
void main ( ) {
    short arr[3] = {10,20,30};
    short *sp;
    sp = arr;
    printf(" *sp = %d\n", *sp );
    sp ++;
    printf(" *sp = %d\n", *sp );
    printf(" sp[1] = %d\n", sp[1] );
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
*sp = 10
*sp = 20
sp[1] = 30
Press any key to continue
```


예제 8-14 : 이차원 배열의 이름이 가지는 의미

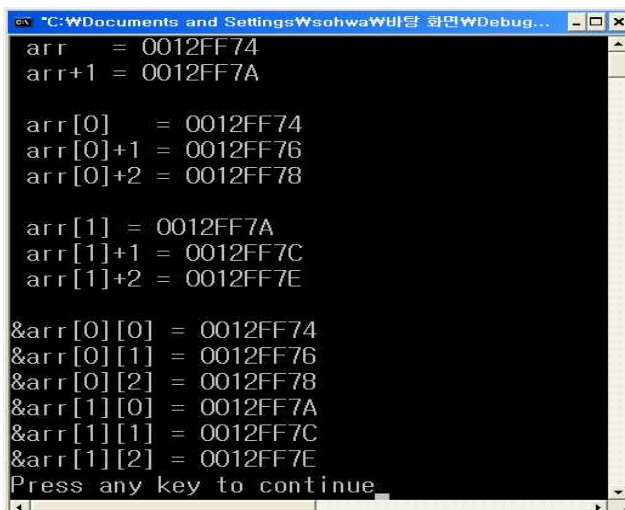
```
#include <stdio.h>
void main ( ) {
    short arr[2][3] = { {10,20,30}, {40,50,60}};
    printf(" arr = %p\n", arr);
    printf(" arr[0] = %p\n", arr[0]);
    printf(" &arr[0][0] = %p\n", &arr[0][0]);
}
```



```
arr = 0012FF74
arr[0] = 0012FF74
&arr[0][0] = 0012FF74
Press any key to continue
```

예제 8-15 : 이차원 배열의 이름 +정수를 했을 경우의 결과

```
#include <stdio.h>
void main ( ) {
    short arr[2][3] = { {10,20,30}, {40,50,60}};
    printf(" arr = %p\n", arr);
    printf(" arr+1 = %p\n", arr+1);
    printf("\n");
    printf(" arr[0] = %p\n", arr[0]);
    printf(" arr[0]+1 = %p\n", arr[0]+1);
    printf(" arr[0]+2 = %p\n", arr[0]+2);
    printf("\n");
    printf(" arr[1] = %p\n", arr[1]);
    printf(" arr[1]+1 = %p\n", arr[1]+1);
    printf(" arr[1]+2 = %p\n", arr[1]+2);
    printf("\n");
    printf("&arr[0][0] = %p\n", &arr[0][0] );
    printf("&arr[0][1] = %p\n", &arr[0][1] );
    printf("&arr[0][2] = %p\n", &arr[0][2] );
    printf("&arr[1][0] = %p\n", &arr[1][0] );
    printf("&arr[1][1] = %p\n", &arr[1][1] );
    printf("&arr[1][2] = %p\n", &arr[1][2] );
}
```



```
arr = 0012FF74
arr+1 = 0012FF7A

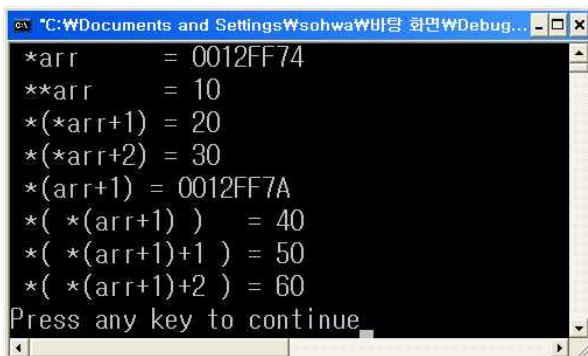
arr[0] = 0012FF74
arr[0]+1 = 0012FF76
arr[0]+2 = 0012FF78

arr[1] = 0012FF7A
arr[1]+1 = 0012FF7C
arr[1]+2 = 0012FF7E

&arr[0][0] = 0012FF74
&arr[0][1] = 0012FF76
&arr[0][2] = 0012FF78
&arr[1][0] = 0012FF7A
&arr[1][1] = 0012FF7C
&arr[1][2] = 0012FF7E
Press any key to continue
```

예제 8-16 : 이차원 배열의 이름을 역참조 했을 때의 결과

```
#include <stdio.h>
void main ( ) {
    short arr[2][3] = { {10,20,30}, {40,50,60}};
    printf(" *arr      = %p\n", *arr);
    printf(" **arr     = %d\n", **arr);
    printf(" *(*arr+1) = %d\n", *(*arr+1) );
    printf(" *(*arr+2) = %d\n", *(*arr+2) );
    printf(" *(arr+1) = %p\n", *(arr+1) );
    printf(" *( *(arr+1) )  = %d\n", *( *(arr+1) ) );
    printf(" *( *(arr+1)+1 ) = %d\n", *( *(arr+1)+1 ) );
    printf(" *( *(arr+1)+2 ) = %d\n", *( *(arr+1)+2 ) );
}
```



```
*C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
*arr      = 0012FF74
**arr     = 10
*(*arr+1) = 20
*(*arr+2) = 30
*(arr+1) = 0012FF7A
*( *(arr+1) )  = 40
*( *(arr+1)+1 ) = 50
*( *(arr+1)+2 ) = 60
Press any key to continue
```

예제 8-17 : 이차원 배열의 이름을 포인터 변수에 저장하는 예제

```
#include <stdio.h>
void main ( ) {
    short arr[2][3] = { {10,20,30}, {40,50,60}};
    short *sp1;      // short의 주소값을 가질 수 있는 포인터 변수가 만들어 진다.
    short *sp2[3];    // short형 포인터 변수가 3개 만들어 진다.
    short (*sp3)[3];  // short형 [3]칸짜리 배열전체를 가리킬 수 있는 sp3가 만들어 진다.
    sp3 = arr;
    printf(" arr   = %p\n", arr);
    printf(" arr+1 = %p\n", arr+1);
    printf(" sp3   = %p\n", sp3 );
    printf(" sp3+1 = %p\n", sp3+1 );
    printf(" sp3[0][0] = %d\n", sp3[0][0] );
    printf(" sp3[0][1] = %d\n", sp3[0][1] );
    printf(" sp3[0][2] = %d\n", sp3[0][2] );
    printf(" sp3[1][0] = %d\n", sp3[1][0] );
    printf(" sp3[1][1] = %d\n", sp3[1][1] );
    printf(" sp3[1][2] = %d\n", sp3[1][2] );
}
```

```

arr = 0012FF74
arr+1 = 0012FF7A
sp3 = 0012FF74
sp3+1 = 0012FF7A
sp3[0][0] = 10
sp3[0][1] = 20
sp3[0][2] = 30
sp3[1][0] = 40
sp3[1][1] = 50
sp3[1][2] = 60
Press any key to continue

```

예제 8-18 : 이차원 배열의 각 행을 포인터 배열에 저장하는 방법

```

#include <stdio.h>
void main ( ) {
    short arr[2][3] = { {10,20,30}, {40,50,60}};
    short *sp[2];
    sp[0] = arr[0];
    sp[1] = arr[1];
    printf(" sp[0][0] = %d\n", sp[0][0] );
    printf(" sp[0][1] = %d\n", sp[0][1] );
    printf(" sp[0][2] = %d\n", sp[0][2] );
    printf(" sp[1][0] = %d\n", sp[1][0] );
    printf(" sp[1][1] = %d\n", sp[1][1] );
    printf(" sp[1][2] = %d\n", sp[1][2] );
}

```

```

sp[0][0] = 10
sp[0][1] = 20
sp[0][2] = 30
sp[1][0] = 40
sp[1][1] = 50
sp[1][2] = 60
Press any key to continue

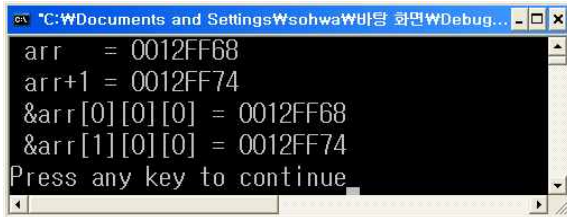
```

예제 8-19 : 3차원 배열의 이름이 가지는 의미

```

#include <stdio.h>
void main ( ) {
    short arr[2][2][3] = { {{10,20,30},{40,50,60}} , {{70,80,90},{100,110,120}} };
    printf(" arr = %p\n", arr);
    printf(" arr+1 = %p\n", arr+1);
    printf(" &arr[0][0][0] = %p\n", &arr[0][0][0]);
    printf(" &arr[1][0][0] = %p\n", &arr[1][0][0]);
}

```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
arr = 0012FF68
arr+1 = 0012FF74
&arr[0][0][0] = 0012FF68
&arr[1][0][0] = 0012FF74
Press any key to continue
```

🔄 문자열

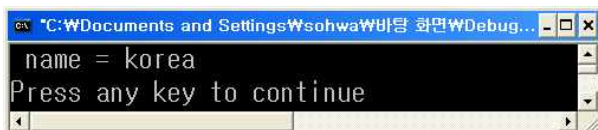
문자열이란 여러개의 문자가 모여 하나의 의미를 가지게 된 것을 말한다. 즉, 문자열은 char의 집합이라 말할 수 있다. 이러한 문자열을 보관하기 위해서는 여러개의 char형 변수가 필요하다. 하지만 문자열은 특성상 길이가 얼마가 될지 알 수 없기 때문에 호출할 때 문제가 발생한다. 따라서 문자열은 메모리에 char가 연달아 나열되어 있는 형태로 보관시킨 뒤 맨 마지막에 NULL문자를 집어 넣어 문자열의 끝을 표시하고 문자열의 첫 번째 글자의 주소값으로 문자열의 위치를 보관한다.

문자열은 char들의 집합임으로 메모리에 char가 연달아 보관될 수 있도록 char형 배열을 만들어 보관시킨다. 따라서 문자열을 저장하려면 반드시 char형 배열을 만들어야 한다. 단, char형 배열의 이름이 첫 번째 원소를 가리키는 포인터 상수임으로 이미 문자열이 보관되어 있거나 문자열 상수의 경우에는 해당 문자열을 호출해 사용하기 위해서는 char형 포인터 변수를 사용할 수 있다. 문자열은 길이가 모두 다르므로 문자열의 끝이 표시하기 위해 항상 마지막에 NULL문자가 들어가야 한다.

문자열은 출력시 puts() 함수를 사용하거나 printf() 함수에서 %s라는 출력서식을 이용해 출력한다. 문자열은 출력되면 시작주소에 들어 있는 문자부터 NULL문자가 나올때까지 연달아 출력된다.

예제 8-20 : char형 배열을 이용해 문자열을 저장하고 출력하는 방법

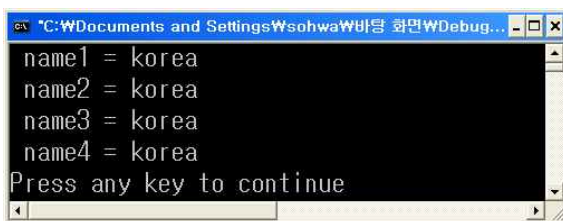
```
#include <stdio.h>
void main ( ) {
    char name[6] = { 'k', 'o', 'r', 'e', 'a', '\0' };
    printf(" name = %s \n", name);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
name = korea
Press any key to continue
```

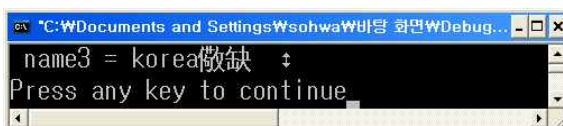
예제 8-21 : char형 배열의 값을 문자열로 초기화 시키는 여러 가지 방법

```
#include <stdio.h>
void main ( ) {
    char name1[6] = { 'k', 'o', 'r', 'e', 'a', '\0' };
    char name2[6] = { 'k', 'o', 'r', 'e', 'a', NULL };
    char name3[6] = "korea";
    char name4[] = "korea";
    printf(" name1 = %s \n", name1);
    printf(" name2 = %s \n", name2);
    printf(" name3 = %s \n", name3);
    printf(" name4 = %s \n", name4);
}
```



예제 8-22 : char형 배열의 값을 문자열로 초기화 시킬때의 주의사항

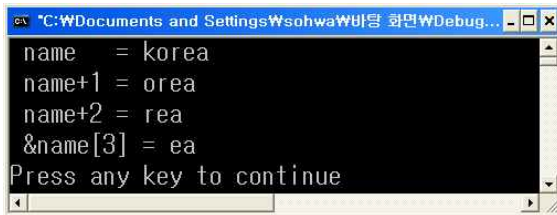
```
#include <stdio.h>
void main ( ) {
    // char name1[5] = { 'k', 'o', 'r', 'e', 'a', '\0' };
    // char name2[5] = "korea";
    // 배열의 크기보다 초기값이 많으면 에러가 발생한다.
    char name3[5] = { 'k', 'o', 'r', 'e', 'a' };
    // 문자열의 마지막에 NULL이 없다면 불완전 문자열이 된다.
    printf(" name3 = %s \n", name3);
}
```



설명 : 불완전 문자열은 마지막에 NULL이 없기 때문에 출력 시 NULL문자가 나올 때 까지 메모리의 data를 모두 출력한다. 때문에 위의 결과처럼 쓰레기 값도 같이 출력되게 된다.

예제 8-23 : 문자열은 시작주소부터 NULL이 나올때까지 출력되는 것을 확인하는 예제

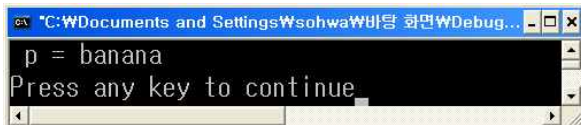
```
#include <stdio.h>
void main ( ) {
    char name[] = "korea";
    printf(" name = %s \n", name);
    printf(" name+1 = %s \n", name+1);
    printf(" name+2 = %s \n", name+2);
    printf(" &name[3] = %s \n", &name[3]);
}
```



```
name = korea
name+1 = orea
name+2 = rea
&name[3] = ea
Press any key to continue
```

예제 8-24 : 포인터를 이용한 문자열의 출력방법

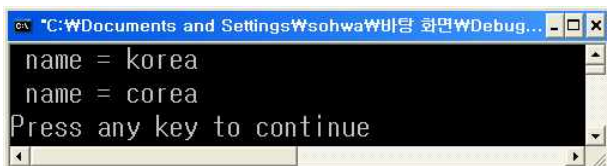
```
#include <stdio.h>
void main ( ) {
    char name[] = "banana";
    char *p = name;
    printf(" p = %s\n", p);
}
```



```
p = banana
Press any key to continue
```

예제 8-25 : 배열안에 있는 문자를 변경하는 방법

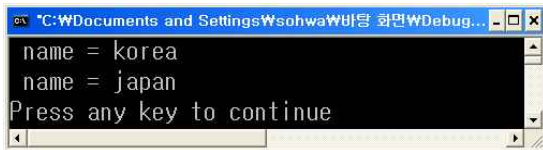
```
#include <stdio.h>
void main ( ) {
    char name[] = "korea";
    printf(" name = %s\n", name);
    name[0] = 'c';
    printf(" name = %s\n", name);
}
```



```
name = korea
name = corea
Press any key to continue
```

예제 8-26 : 배열안에 있는 문자열을 변경하는 방법

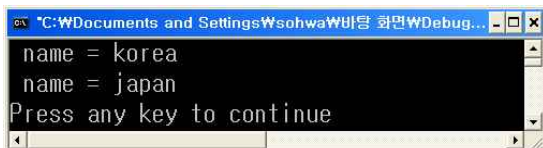
```
#include <stdio.h>
void main ( ) {
    char name[] = "korea";
    printf(" name = %s\n", name);
    name[0] = 'j';
    name[1] = 'a';
    name[2] = 'p';
    name[3] = 'a';
    name[4] = 'n';
    printf(" name = %s\n", name);
}
```



```
name = korea
name = japan
Press any key to continue
```

예제 8-27 : strcpy함수를 이용해 문자열을 변경하는 방법

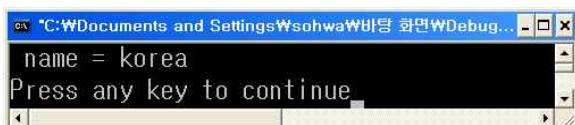
```
#include <stdio.h>
#include <string.h>          // strcpy함수를 사용하기 위한 헤더파일
void main ( ) {
    char name[] = "korea";
    printf(" name = %s\n", name);
    strcpy( name, "japan" );    // name에 "japan"을 복사해 넣는다.
    printf(" name = %s\n", name);
}
```



```
name = korea
name = japan
Press any key to continue
```

예제 8-28 : char 형 포인터를 이용해 문자열의 위치를 저장하는 방법

```
#include <stdio.h>
void main ( ) {
    char *name = "korea";
    printf(" name = %s\n", name);
}
```

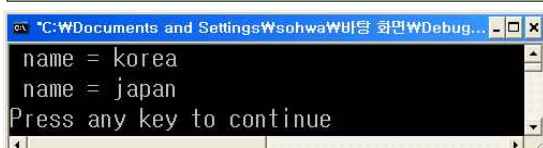


```
name = korea
Press any key to continue
```

예제 8-29 : char 형 포인터는 문자열의 위치만을 가리킴으로 그 안의 data를 변경 할 수 없다는 것을 보여주는 예제

```
#include <stdio.h>
void main ( ) {
    char *name = "korea";
    printf(" name = %s\n", name);
    // *name = 'c';          // name이 가리키는 곳은 할당받은 메모리가 아님으로 data를
                             // 넣으면 에러가 발행한다.

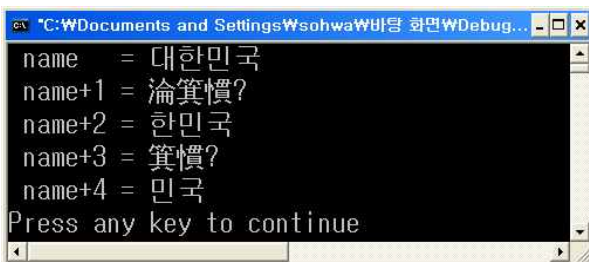
    name = "japan";          // name은 변수임으로 다른 주소값을 가질수 있다.
    printf(" name = %s\n", name);
}
```



```
name = korea
name = japan
Press any key to continue
```


예제 8-30 : 배열과 포인터가 문자열을 보관할 때의 차이점을 보여주는 예제

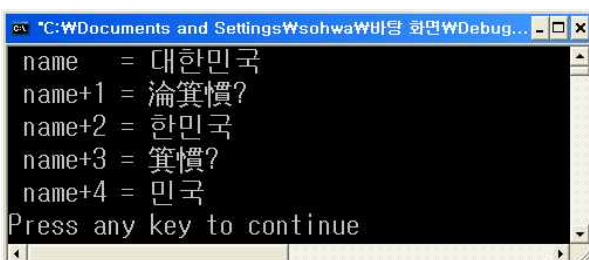
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char name1[] = "korea";
    char *name2 = "korea";
    printf(" name1 = %s\n", name1);
    printf(" name2 = %s\n", name2);
    *name1 = 'c';
    // *name2 = 'c';          // 할당받지 않은곳에 data를 넣었음으로 에러
    printf(" name1 = %s\n", name1);
    printf(" name2 = %s\n", name2);
    strcpy( name1, "china");
    // strcpy( name2, "china"); // 할당받지 않은곳에 data를 넣었음으로 에러
    printf(" name1 = %s\n", name1);
    printf(" name2 = %s\n", name2);
    // name1 = "spain";      // 에러 : 배열의 이름은 상수임으로 다른주소를 보관할 수 없다.
    name2 = "spain";
    printf(" name1 = %s\n", name1);
    printf(" name2 = %s\n", name2);
}
```



```
name = 대한민국
name+1 = 淪箕慣?
name+2 = 한민국
name+3 = 箕慣?
name+4 = 민국
Press any key to continue
```

예제 8-31 : 한글은 한 글자에 2byte라는 것을 보여주는 예제

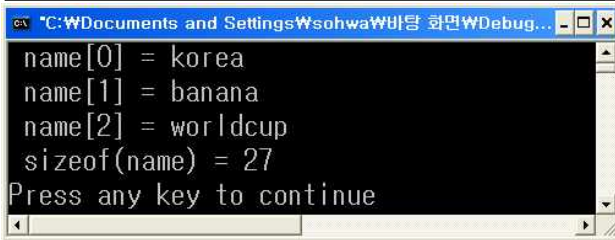
```
#include <stdio.h>
void main ( ) {
    char *name = "대한민국";
    printf(" name = %s\n", name);
    printf(" name+1 = %s\n", name+1);
    printf(" name+2 = %s\n", name+2);
    printf(" name+3 = %s\n", name+3);
    printf(" name+4 = %s\n", name+4);
}
```



```
name = 대한민국
name+1 = 淪箕慣?
name+2 = 한민국
name+3 = 箕慣?
name+4 = 민국
Press any key to continue
```

예제 8-32 : char형 이차원 배열에 문자열을 보관하는 방법

```
#include <stdio.h>
void main ( ) {
    char name[3][9] = {"korea", "banana", "worldcup" };
    printf(" name[0] = %s\n", name[0]);
    printf(" name[1] = %s\n", name[1]);
    printf(" name[2] = %s\n", name[2]);
    printf(" sizeof(name) = %d\n", sizeof(name) );
}
```

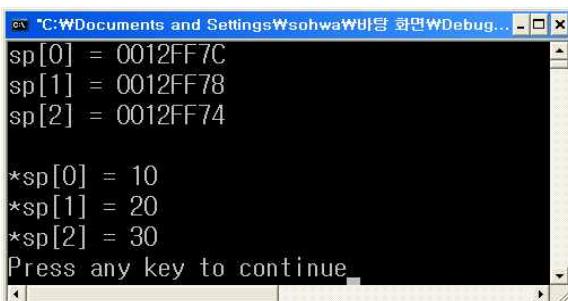


🔄 포인터 배열

포인터 변수도 변수임으로 배열로 만들 수 있다. 포인터 변수를 배열로 만들면 배열의 원소 하나하나가 포인터 변수가 된다. 또한 배열의 이름은 배열의 첫 번째 원소를 가리키는 포인터 상수임으로 포인터 배열의 이름은 중첩포인터가 된다.

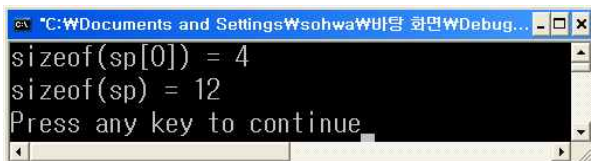
예제 8-33 : 포인터 배열을 만드는 방법

```
#include <stdio.h>
void main ( ) {
    short a = 10, b = 20, c = 30;
    short *sp[3];    // 메모리에 short형 포인터변수가 세 개 만들어 진다.
    sp[0] = &a;
    sp[1] = &b;
    sp[2] = &c;
    printf("sp[0] = %p\n", sp[0]);    // sp[0]에 들어 있는 a의 주소값이 나온다.
    printf("sp[1] = %p\n", sp[1]);    // sp[1]에 들어 있는 b의 주소값이 나온다.
    printf("sp[2] = %p\n", sp[2]);    // sp[2]에 들어 있는 c의 주소값이 나온다.
    printf("\n");
    printf("*sp[0] = %d\n", *sp[0]);    // sp[0]이 가리키는 a의 값 10이 나온다.
    printf("*sp[1] = %d\n", *sp[1]);    // sp[1]이 가리키는 b의 값 20이 나온다.
    printf("*sp[2] = %d\n", *sp[2]);    // sp[2]가 가리키는 c의 값 30이 나온다.
}
```



예제 8-34 : 포인터 배열에 있는 각 원소의 크기와 배열 전체의 크기

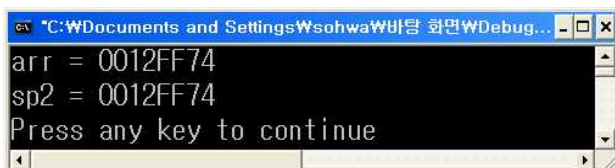
```
#include <stdio.h>
void main ( ) {
    short *sp[3];    // 메모리에 short형 포인터변수가 세 개 만들어 진다.
    printf("sizeof(sp[0]) = %d Wn", sizeof(sp[0]) );
    printf("sizeof(sp) = %d Wn", sizeof(sp) );
}
```



```
C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
sizeof(sp[0]) = 4
sizeof(sp) = 12
Press any key to continue
```

예제 8-35 : 포인터 배열의 이름이 어떠한 자료형인지 알아보는 예제

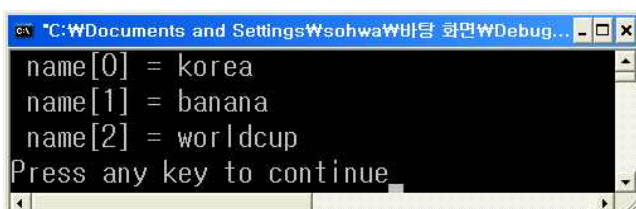
```
#include <stdio.h>
void main ( ) {
    short *arr[3];    // 메모리에 short형 포인터변수가 세 개 만들어 진다.
    short *sp1;
    short **sp2;
    // sp1 = arr;      // 에러
    sp2 = arr;
    printf("arr = %pWn", arr);
    printf("sp2 = %pWn", sp2);
}
```



```
C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
arr = 0012FF74
sp2 = 0012FF74
Press any key to continue
```

예제 8-36 : 포인터 배열을 이용한 문자열의 저장

```
#include <stdio.h>
void main ( ) {
    char *name[3] = {"korea", "banana", "worldcup" };
    printf(" name[0] = %sWn", name[0] );
    printf(" name[1] = %sWn", name[1] );
    printf(" name[2] = %sWn", name[2] );
}
```



```
C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
name[0] = korea
name[1] = banana
name[2] = worldcup
Press any key to continue
```

예제 8-37 : 포인터 배열을 이차원 배열처럼 활용하는 예제

```
#include <stdio.h>
void main ( ) {
    char *name[3] = {"korea", "banana", "worldcup" };
    printf(" name[0] = %s\n", name[0] );
    printf(" name[1] = %s\n", name[1] );
    printf(" name[2] = %s\n", name[2] );
    printf("\n");
    printf(" name[0][0] = %d\n", name[0][0] );
    printf(" name[0][1] = %d\n", name[0][1] );
    printf("\n");
    printf(" name[1][0] = %d\n", name[1][0] );
    printf(" name[1][1] = %d\n", name[1][1] );
    printf("\n");
    printf(" name[2][0] = %d\n", name[2][0] );
    printf(" name[2][1] = %d\n", name[2][1] );
}
```

```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
name[0] = korea
name[1] = banana
name[2] = worldcup

name[0][0] = k
name[0][1] = o

name[1][0] = b
name[1][1] = a

name[2][0] = w
name[2][1] = o
Press any key to continue
```

예제 8-38 : 포인터 변수의 활용

```
#include <stdio.h>
void main ( ) {
    char *name[3] = {"korea", "banana", "worldcup" };
    char **cp = name;
    printf(" *cp      = %s\n", *cp);
    printf(" *cp+1    = %s\n", *cp+1);
    printf(" *(cp+1)   = %s\n", *(cp+1));
    printf(" *(cp+1)+1 = %s\n", *(cp+1)+1);
    printf("\n");
    printf(" **cp      = %c\n", **cp );
    printf(" **(cp+1)  = %c\n", **(cp+1) );
    printf(" *(*cp+1) = %c\n", *(*cp+1) );
    printf(" **cp+1    = %c\n", **cp+1 );
}
```

```

C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
*cp      = korea
*cp+1    = orea
*(cp+1)  = banana
*(cp+1)+1 = anana

**cp     = k
**(cp+1) = b
*(*cp+1) = o
**cp+1   = l
Press any key to continue

```

void 포인터

포인터 변수를 만들면서 자료형을 void로 선택하면 모든 형태의 주소값을 보관할 수 있는 void 포인터 변수를 만들 수 있다. void 포인터는 자료형이 없기 때문에 모든 형태의 주소값을 보관할 수 있지만 역참조를 통해 data를 꺼내오는 것은 불가능하다. 만약 void 포인터가 가리키고 있는 값을 역참조를 통해 꺼내오려면 void포인터를 강제 형변환 한 뒤 역참조 해야 한다.

예제 8-39 : void 포인터 변수의 의미

```

#include <stdio.h>
void main ( ) {
    void *vp;
    int a;          int *ip;
    char b;         char name[10];
    double c;       double **dp;
    vp = &a;
    vp = &ip;
    vp = &b;
    vp = &name;
    vp = &c;
    vp = &dp;
}

```

예제 8-40 : void 포인터를 역참조 하는 예제 : 에러발생

```

#include <stdio.h>
void main ( ) {
    void *vp;
    int a = 10;
    vp = &a;
    // printf(" *vp = %d\n", *vp); // vp는 자료형이 없기 때문에 역참조하면 에러가
    //                             // 발생한다.

    printf(" *(int *)vp = %d\n", *(int *)vp);
    // vp를 int *로 강제 형변환 했음으로 역참조하면
    // 결과가 int로 나오게 된다.
}

```

```

C:\Documents and Settings\WsohwaW바탕 화면\WDebug...
*(int *)vp = 10
Press any key to continue

```

예제 8-41 : void 포인터를 활용하는 예제

```

#include <stdio.h>
void prn ( void *vp, int x) {
    if( x == 1 ) {
        printf(" %d Wn", *(int *)vp );
    }
    else if( x == 2 ) {
        printf(" %lf Wn", *(double *)vp );
    }
}
void main ( ) {
    int a = 10;
    double b = 3.14;
    prn( &a, 1 );
    prn( &b, 2 );
}

```

```

C:\Documents and Settings\WsohwaW바탕 화면\WDebug...
10
3.140000
Press any key to continue

```

상수형 포인터 변수와 포인터 상수

const 키워드를 붙여서 만들어지는 상수의 주소값을 보관하는 포인터 변수를 만들기 위해서는 포인터 변수 앞에도 const 키워드를 붙여서 변수를 만들어야 한다. 즉,

const 자료형 상수명 = 상수값; 과 같이 변수를 만들면

const 자료형 *포인터변수명 = &상수명; 의 형태로 주소값을 보관 해야 한다.

또한 포인터 변수 자체를 상수로 만들고 싶다면

자료형 * const 포인터변수명 = &변수명;

의 형태로 포인터를 만들면 만들어지는 포인터 변수는 상수가 된다.

예제 8-42 : 상수에 대한 포인터 변수를 만드는 예제

```

#include <stdio.h>
void main ( ) {
    const int a = 10;        // a는 상수임으로 값을 변경 할 수 없다.
    const int b = 20;        // b는 상수임으로 값을 변경 할 수 없다.
    const int *ip = &a;      // a의 주소값을 보관하는 ip는 포인터 변수이다.
    printf(" *ip = %dWn", *ip );
    ip = &b;
    printf(" *ip = %dWn", *ip );
}

```

```

C:\Documents and Settings\sohwa\바탕 화면\WDebug...
*ip = 10
*ip = 20
Press any key to continue

```

예제 8-43 : 포인터를 상수로 만드는 예제

```

#include <stdio.h>
void main ( ) {
    int a = 10;
    int b = 20;
    int * const ip = &a;    // a의 주소값을 보관하는 ip는 포인터 상수가 된다.
    printf(" *ip = %d\n", *ip );
    // ip = &b;    // 에러 : ip는 상수임으로 값을 변경 할 수 없다.
}

```

```

C:\Documents and Settings\sohwa\바탕 화면\WDebug...
*ip = 10
Press any key to continue

```

예제 8-44 : 상수에 대한 포인터 상수를 만드는 예제

```

#include <stdio.h>
void main ( ) {
    const int a = 10;
    const int b = 20;
    const int * const ip = &a; // a도 상수이고 a의 주소값을 보관하는 ip도 상수가 된다.
    printf(" *ip = %d\n", *ip );
    // *ip = 20;    // 에러 : ip가 가리키는 a는 상수임으로 역참조를 통해 값을 변경
                    // 할 수 없다.
    // ip = &b;    // 에러 : ip는 상수임으로 다른 주소값을 보관 할 수 없다.
}

```

```

C:\Documents and Settings\sohwa\바탕 화면\WDebug...
*ip = 10
Press any key to continue

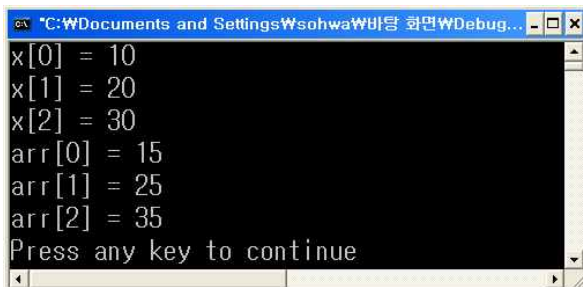
```

배열을 매개변수로 주고 리턴 받는 방법

배열을 함수의 매개변수로 주기 위해서는 배열의 이름을 매개변수 주는 부분에 쓰면 된다. 배열의 이름은 배열 안에 있는 첫 번째 원소의 주소값 임으로 포인터 이고 따라서 배열을 매개변수로 받아 오는 함수는 배열의 형태 또는 포인터 변수를 이용해 배열을 매개변수로 받아 올 수 있다. 단, 함수에 전달되는 것은 배열의 주소값 임으로 함수 호출의 형태 중 call by adress가 된다. 또한 함수 내에서 만든 배열은 함수가 종료되면 사라짐으로 리턴을 할 수 없다.

예제 8-45 : 일차원 배열을 매개변수로 주는 방법 (배열로 받는다.)

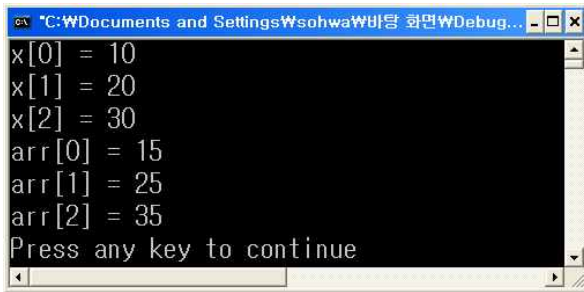
```
#include <stdio.h>
void func( int x[] ) {
    printf("x[0] = %d\n", x[0]);
    printf("x[1] = %d\n", x[1]);
    printf("x[2] = %d\n", x[2]);
    x[0] = 15;
    x[1] = 25;
    x[2] = 35;
}
void main ( ) {
    int arr[3] = { 10,20,30 };
    func ( arr );
    printf("arr[0] = %d\n", arr[0]);
    printf("arr[1] = %d\n", arr[1]);
    printf("arr[2] = %d\n", arr[2]);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
x[0] = 10
x[1] = 20
x[2] = 30
arr[0] = 15
arr[1] = 25
arr[2] = 35
Press any key to continue
```

예제 8-46 : 일차원 배열을 매개변수로 주는 방법 (포인터로 받는다.)

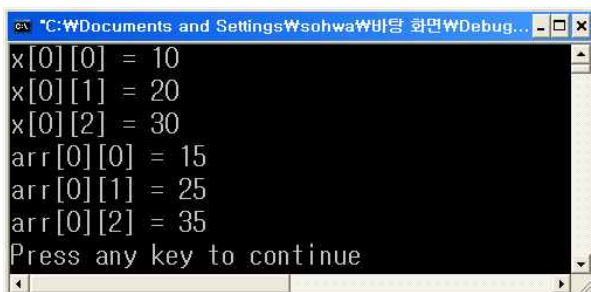
```
#include <stdio.h>
void func( int *x ) {
    printf("x[0] = %d\n", x[0]);
    printf("x[1] = %d\n", x[1]);
    printf("x[2] = %d\n", x[2]);
    x[0] = 15;
    x[1] = 25;
    x[2] = 35;
}
void main ( ) {
    int arr[3] = { 10,20,30 };
    func ( arr );
    printf("arr[0] = %d\n", arr[0]);
    printf("arr[1] = %d\n", arr[1]);
    printf("arr[2] = %d\n", arr[2]);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
x[0] = 10
x[1] = 20
x[2] = 30
arr[0] = 15
arr[1] = 25
arr[2] = 35
Press any key to continue
```

예제 8-47 : 이차원 배열을 매개변수로 주는 방법 (배열로 받는다.)

```
#include <stdio.h>
void func( int x[][3] ) {
    printf("x[0][0] = %d\n", x[0][0]);
    printf("x[0][1] = %d\n", x[0][1]);
    printf("x[0][2] = %d\n", x[0][2]);
    x[0][0] = 15;
    x[0][1] = 25;
    x[0][2] = 35;
}
void main ( ) {
    int arr[2][3] = { {10,20,30}, {40,50,60} };
    func ( arr );
    printf("arr[0][0] = %d\n", arr[0][0]);
    printf("arr[0][1] = %d\n", arr[0][1]);
    printf("arr[0][2] = %d\n", arr[0][2]);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
x[0][0] = 10
x[0][1] = 20
x[0][2] = 30
arr[0][0] = 15
arr[0][1] = 25
arr[0][2] = 35
Press any key to continue
```

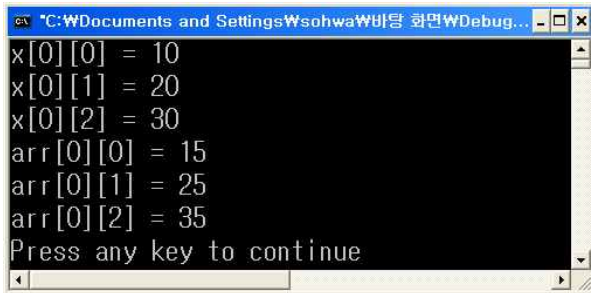
예제 8-48 : 이차원 배열을 매개변수로 주는 방법 (포인터로 받는다.)

```
#include <stdio.h>
void func( int (*x)[3] ) {
    printf("x[0][0] = %d\n", x[0][0]);
    printf("x[0][1] = %d\n", x[0][1]);
    printf("x[0][2] = %d\n", x[0][2]);
    x[0][0] = 15;
    x[0][1] = 25;
    x[0][2] = 35;
}
```

```

void main ( ) {
    int arr[2][3] = { {10,20,30}, {40,50,60} };
    func ( arr );
    printf("arr[0][0] = %d\n", arr[0][0]);
    printf("arr[0][1] = %d\n", arr[0][1]);
    printf("arr[0][2] = %d\n", arr[0][2]);
}

```



```

C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
x[0][0] = 10
x[0][1] = 20
x[0][2] = 30
arr[0][0] = 15
arr[0][1] = 25
arr[0][2] = 35
Press any key to continue

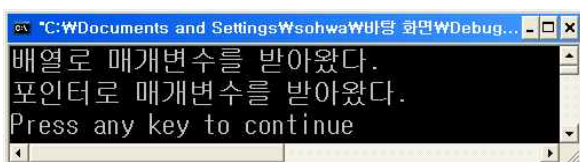
```

예제 8-49 : 삼차원 배열을 매개변수로 주는 방법

```

#include <stdio.h>
void func1( int x[][2][3] ) {
    printf("배열로 매개변수를 받아왔다.\n");
}
void func2( int (*x)[2][3] ) {
    printf("포인터로 매개변수를 받아왔다.\n");
}
void main ( ) {
    int arr[2][2][3] = { {{10,20,30}, {40,50,60}}, {{10,20,30}, {40,50,60}} };
    func1 ( arr );
    func2 ( arr );
}

```



```

C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
배열로 매개변수를 받아왔다.
포인터로 매개변수를 받아왔다.
Press any key to continue

```

예제 8-50 : 배열을 리턴했을 경우의 결과를 살펴보는 예제

```

#include <stdio.h>
int * func( ) {
    int arr[3] = {10, 20, 30 };
    return arr;
}
void main ( ) {
    int *ip;
    ip = func( );
    printf(" ip[0] = %d\n", ip[0]);
}

```

```
printf(" ip[1] = %d\n", ip[2]);
printf(" ip[1] = %d\n", ip[2]);
}
```

```
C:\WDocuments and Settings\Wshwa\바탕 화면\WDebug...
ip[0] = 10
ip[1] = 4241648
ip[1] = 4241668
Press any key to continue
```

8-2. Maker Function

🔄 Maker Function

메이커 function이란 컴파일러를 만든 회사에서 제공하는 함수를 말한다. 해당 메이커 function들은 헤더 파일에 포함되어 있으므로 해당 헤더파일을 include해야 사용할 수 있다.

자주 사용하는 메이커 평선이 있는 헤더파일

- ① string.h : 문자열 처리와 관련된 함수가 들어 있다.
- ② math.h : 수학적 계산과 관련된 함수가 들어 있다.
- ③ stdlib.h : 자주 사용하는 유용한 함수가 들어 있다.
- ④ ctype.h : 문자 처리와 관련된 함수가 들어 있다.
- ⑤ time.h : 시간과 관련된 함수가 들어 있다.

string.h에 포함되어 있는 함수 중 자주 사용하는 함수

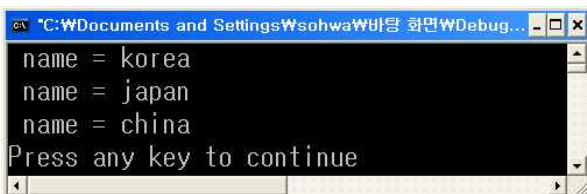
- ① strcpy 함수 :
strcpy(문자열1, 문자열2) 의 형태로 사용되며 매개변수로 받아간 문자열2를 문자열1에 복사해준다. 따라서 문자열1은 반드시 할당된 메모리를 가리키고 있어야 한다.
- ② strcat 함수
strcat(문자열1, 문자열2) 의 형태로 사용되며 매개변수로 받아간 문자열2를 문자열1의 뒤쪽으로 합친 후 문자열 1의 주소를 리턴한다. 따라서 문자열1은 반드시 할당된 메모리를 가리키고 있어야 하며 그 크기가 문자열 1 + 문자열 2보다 크거나 같아야 한다.
- ③ strcmp 함수
strcmp(문자열1, 문자열2) 의 형태로 사용되며 매개변수로 받아간 문자열1과 문자열2를 비교해 어떤 문자열의 크기가 더 큰지 비교해 준다. 문자열1이 더 크다면 1, 문자열2가 더 크다면 -1, 두 문자열이 같다면 0이 함수의 실행 결과로 나온다.
- ④ strlen 함수
strlen(문자열1) 의 형태로 사용되며 매개변수로 받아간 문자열1 안에 있는 문자의 개수를 세어 리턴한다. 예를 들어 strlen("korea") 하면 실행 결과로 5가 나오게 된다. 앞에서 알 수 있듯이 NULL문자는 문자의 개수로 포함시키지 않는다.

⑤ strchr 함수

strchr(문자열, '문자') 의 형태로 사용되며 첫 번째 매개변수로 받아간 문자열 안에 있는 두 번째 문자가 있는지 검색하여 있다면 그 문자의 주소 값을 리턴해 준다.

예제 8-51 : strcpy 함수의 사용 예

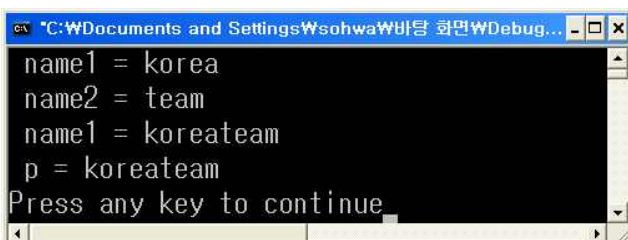
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char name[6]= "korea";
    printf(" name = %s \n", name);
    strcpy( name, "japan");
    printf(" name = %s \n", name);
    strcpy( name, "china");
    printf(" name = %s \n", name);
}
```



```
C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
name = korea
name = japan
name = china
Press any key to continue
```

예제 8-52 : strcat 함수의 사용 예

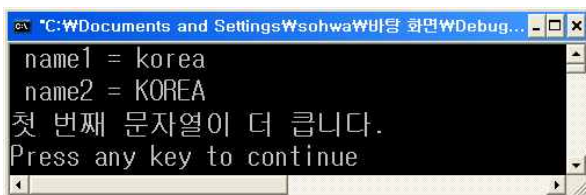
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char name1[11]= "korea";
    char name2[5]= "team";
    char *p;
    printf(" name1 = %s \n", name1);
    printf(" name2 = %s \n", name2);
    p = strcat( name1, name2);
    printf(" name1 = %s \n", name1);
    printf(" p = %s \n", p);
}
```



```
C:\WDocuments and Settings\Wsohwa\바탕 화면\WDebug...
name1 = korea
name2 = team
name1 = koreateam
p = koreateam
Press any key to continue
```

예제 8-53 : strcmp 함수의 사용 예 - 1

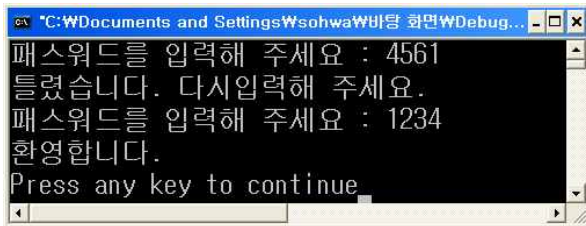
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    int x;
    char name1[6]= "korea";
    char name2[6]= "KOREA";
    printf(" name1 = %s \n", name1);
    printf(" name2 = %s \n", name2);
    x = strcmp( name1, name2);
    if( x == 0 ) {
        printf("두 문자열은 같습니다.\n");
    }
    else if ( x == 1 ) {
        printf("첫 번째 문자열이 더 큼니다.\n");
    }
    else if ( x == -1 ) {
        printf("두 번째 문자열이 더 큼니다.\n");
    }
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
name1 = korea
name2 = KOREA
첫 번째 문자열이 더 큼니다.
Press any key to continue
```

예제 8-54 : strcmp 함수의 사용 예 - 2

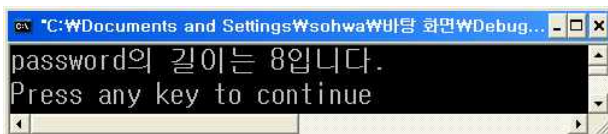
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char *password = "1234";
    char inp[10];
    while(1) {
        printf("패스워드를 입력해 주세요 : ");
        scanf("%s", inp);
        if( strcmp(inp, password) == 0 ) {
            printf("환영합니다.\n");
            break;
        }
        printf("틀렸습니다. 다시입력해 주세요.\n");
    }
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
패스워드를 입력해 주세요 : 4561
틀렸습니다. 다시입력해 주세요.
패스워드를 입력해 주세요 : 1234
환영합니다.
Press any key to continue
```

예제 8-55 : strlen 함수의 사용 예 - 1

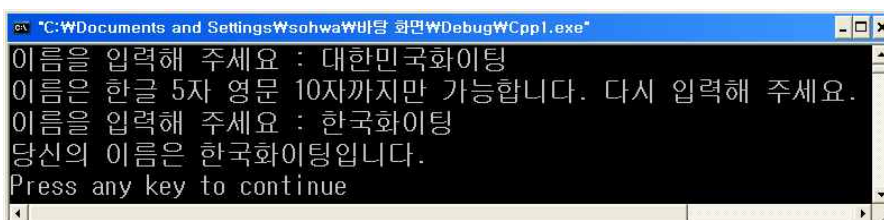
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char *password = "12345678";
    printf("password의 길이는 %d입니다.\n", strlen(password) );
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug...
password의 길이는 8입니다.
Press any key to continue
```

예제 8-56 : strlen 함수의 사용 예 - 2

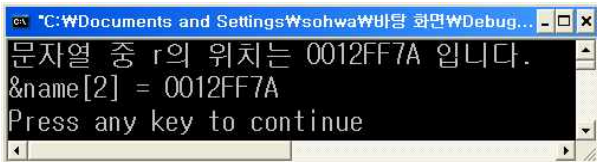
```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char name[100];
    while(1) {
        printf("이름을 입력해 주세요 : ");
        scanf("%s", name);
        if ( strlen(name) <= 10 ) {
            break;
        }
        printf("이름은 한글 5자 영문 10자까지만 가능합니다. 다시 입력해 주세요. \n");
    }
    printf("당신의 이름은 %s입니다.\n", name);
}
```



```
C:\Documents and Settings\sohwa\바탕 화면\WDebug\WCpl.exe
이름을 입력해 주세요 : 대한민국화이팅
이름은 한글 5자 영문 10자까지만 가능합니다. 다시 입력해 주세요.
이름을 입력해 주세요 : 한국화이팅
당신의 이름은 한국화이팅입니다.
Press any key to continue
```


예제 8-57 : strchr 함수의 사용 예

```
#include <stdio.h>
#include <string.h>
void main ( ) {
    char name[6] = "korea";
    printf("문자열 중 r의 위치는 %p 입니다.\n", strchr(name, 'r') );
    printf("&name[2] = %p \n", &name[2]);
}
```

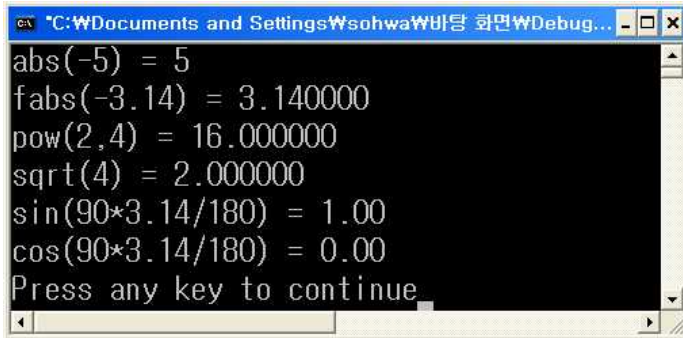


math.h에 포함되어 있는 함수 중 자주 사용하는 함수

- ① pow 함수 :
pow(정수1, 정수2) 의 형태로 사용되며 정수1의 정수2 제곱 값을 구해 준다.
- ② abs 함수 :
pow(정수) 의 형태로 사용되며 매개변수로 받아간 정수의 절대값을 구하여 리턴해 준다.
- ③ fabs 함수 :
fabs(실수) 의 형태로 사용되며 매개변수로 받아간 실수의 절대값을 구하여 리턴해 준다.
- ④ sqrt 함수 :
sqrt(정수) 의 형태로 사용되며 정수의 제곱근(루트) 값을 구하여 리턴해 준다.
- ⑤ sin 함수 :
sin(라디안) 의 형태로 사용되며 라디안에 해당하는 sign값을 구해 리턴해 준다. 각도를 라디안으로 변경하려면 다음의 공식을 참고한다. 라디안 = x도 * 3.14 / 180
- ⑥ cos 함수 :
cos(라디안) 의 형태로 사용되며 라디안에 해당하는 sign값을 구해 리턴해 준다.

예제 8-58 : math.h 안에 있는 함수들의 사용 예

```
#include <stdio.h>
#include <math.h>
void main ( ) {
    printf("abs(-5) = %d \n", abs(-5) );
    printf("fabs(-3.14) = %lf \n", fabs(-3.14) );
    printf("pow(2,4) = %lf \n", pow(2,4) );
    printf("sqrt(4) = %lf \n", sqrt(4) );
    printf("sin(90*3.14/180) = %.2f \n", sin(90*3.14/180) );
    printf("cos(90*3.14/180) = %.2f \n", cos(90*3.14/180) );
}
```



```
abs(-5) = 5
fabs(-3.14) = 3.140000
pow(2,4) = 16.000000
sqrt(4) = 2.000000
sin(90*3.14/180) = 1.00
cos(90*3.14/180) = 0.00
Press any key to continue
```

stdlib.h에 포함되어 있는 함수 중 자주 사용하는 함수

① atoi 함수 :

atoi("문자열1")의 형태로 사용되며 atoi는 ascii to integer의 약자이다. 즉, atoi함수는 매개변수로 받아간 문자열을 int로 변환한 값을 리턴해 준다.

② atol 함수 :

atol("문자열1")의 형태로 사용되며 atol는 ascii to long의 약자이다. 즉, atol함수는 매개변수로 받아간 문자열을 long으로 변환한 값을 리턴해 준다.

③ atof 함수 :

atof("문자열1")의 형태로 사용되며 atof는 ascii to float의 약자이다. 즉, atof함수는 매개변수로 받아간 문자열을 float으로 변환한 값을 리턴해 준다.

④ system 함수 :

system("명령어")의 형태로 사용되며 매개변수로 받아간 도스 명령어를 실행할 수 있게 해준다. 자주 사용하는 명령어에는 cls (화면클리어), dir (폴더내의 파일목록 출력), cd 경로명 (경로변경), md 폴더명 (폴더를 신규로 생성) 등이 있다.

⑤ rand 함수 :

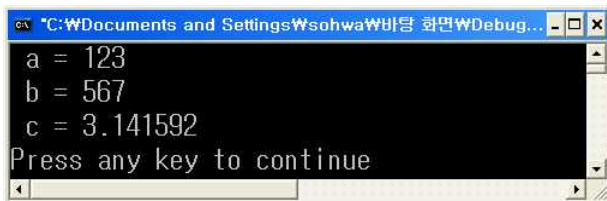
rand()의 형태로 사용되며 0~32767까지의 숫자중 하나를 랜덤으로 리턴해 준다. 단, 해당 숫자를 섞어 놓고 순서대로 하나씩 꺼내주기 때문에 프로그램 종료 후 다시 rand함수가 실행되면 처음 실행 시와 같은 값이 리턴되며 32768회 동안 랜덤수를 가져온다면 겹치는 숫자가 나오지 않는다.

⑥ srand(정수) :

srand(정수)의 형태로 사용되며 매개변수로 받아간 정수만큼 rand함수의 실행 결과를 섞어 준다. 정수가 일정하면 항상 같은 횟수만큼 섞이기 때문에 결과도 같아지게 된다. 따라서 현재 시간을 구해주는 함수를 이용해 srand(time(NULL)); (현재 시간을 정수로 구해와 그 시간만큼 섞는다.) 과 같은 형태로 사용한다. 프로그램 시작 시 한번만 실행하면 된다.

예제 8-58 : 변환함수들의 사용 예

```
#include <stdio.h>
#include <stdlib.h>
void main ( ) {
    int a;
    long b;
    double c;
    a = atoi("123");
    b = atol("567");
    c = atof("3.141592");
    printf(" a = %d \n", a);
    printf(" b = %d \n", b);
    printf(" c = %lf \n", c);
}
```



```
C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
a = 123
b = 567
c = 3.141592
Press any key to continue
```

예제 8-59 : system 함수의 사용한 상자 이동 프로그램

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main ( ) {
    char in;
    int x = 5, i;
    while(1) {
        for(i = 0; i<=10; i++) {
            if( i == x )
                printf("■");
            else
                printf("□");
        }
        printf("\n");
        if( kbhit( ) ) {
            in = getch( );
            if( in == 75 )
                x--;
            else if ( in == 77 )
                x++;
            else if ( in == 27 )
```

```

        break;
    }
    system("cls");
}
}

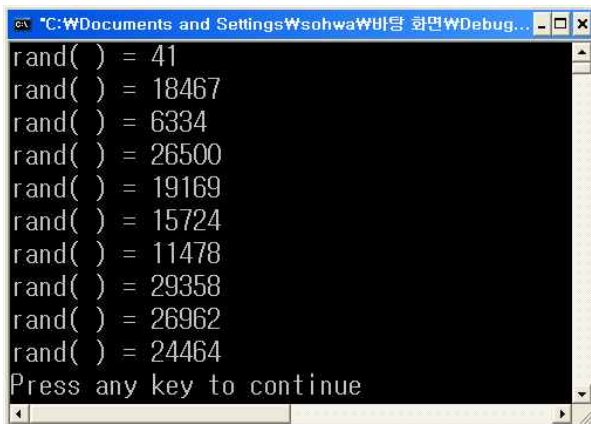
```

예제 8-60 : rand() 함수의 사용방법

```

#include <stdio.h>
#include <stdlib.h>
void main ( ) {
    int a;
    for( a = 1; a <=10 ;a++) {
        printf("rand( ) = %d\n", rand( ) );
    }
}

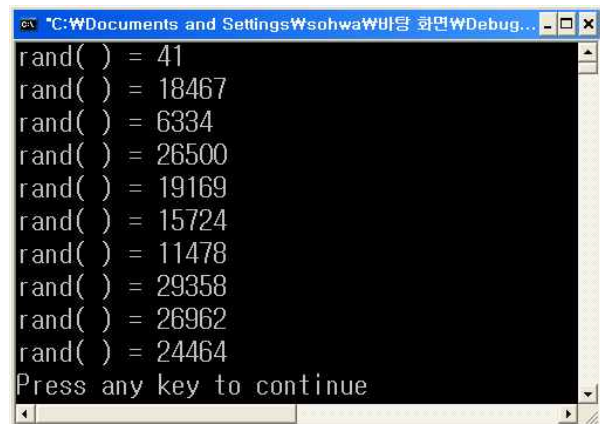
```



```

C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
rand( ) = 41
rand( ) = 18467
rand( ) = 6334
rand( ) = 26500
rand( ) = 19169
rand( ) = 15724
rand( ) = 11478
rand( ) = 29358
rand( ) = 26962
rand( ) = 24464
Press any key to continue

```



```

C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
rand( ) = 41
rand( ) = 18467
rand( ) = 6334
rand( ) = 26500
rand( ) = 19169
rand( ) = 15724
rand( ) = 11478
rand( ) = 29358
rand( ) = 26962
rand( ) = 24464
Press any key to continue

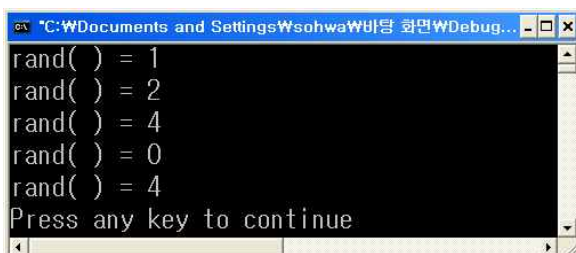
```

예제 8-61 : rand() 함수를 이용해 1~5사이의 랜덤수를 구하는 방법

```

#include <stdio.h>
#include <stdlib.h>
void main ( ) {
    int a;
    for( a = 1; a <=5 ;a++) {
        printf("rand( ) = %d\n", rand( )%5 );
    }
}

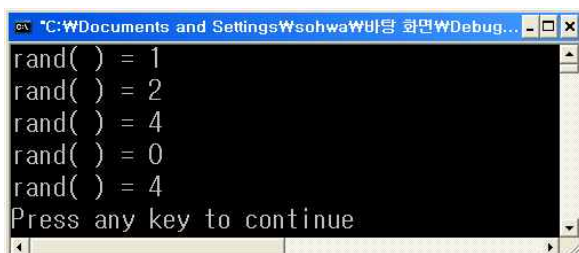
```



```

C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
rand( ) = 1
rand( ) = 2
rand( ) = 4
rand( ) = 0
rand( ) = 4
Press any key to continue

```



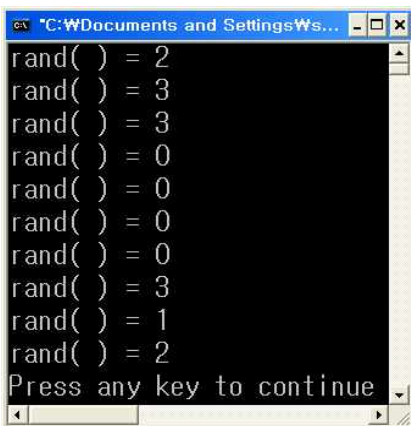
```

C:\Documents and Settings\Wsohwa\바탕 화면\WDebug...
rand( ) = 1
rand( ) = 2
rand( ) = 4
rand( ) = 0
rand( ) = 4
Press any key to continue

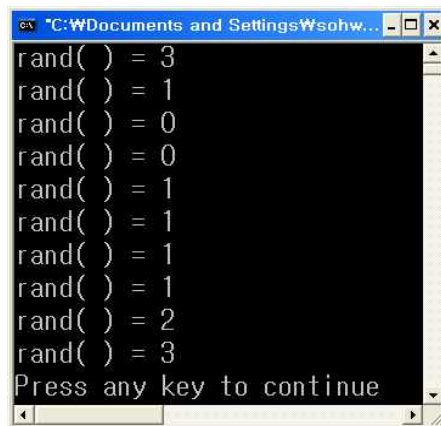
```

예제 8-62 : srand() 함수를 이용해 항상 새로운 랜덤수가 나오게 만든 예제

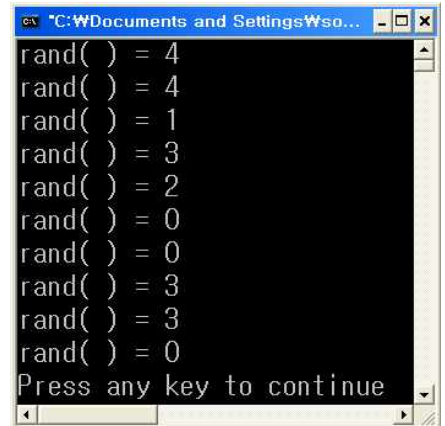
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main ( ) {
    int a;
    srand( time(NULL) );
    for( a = 1; a <=10 ;a++) {
        printf("rand( ) = %d\n", rand( )%5 );
    }
}
```



```
rand( ) = 2
rand( ) = 3
rand( ) = 3
rand( ) = 0
rand( ) = 0
rand( ) = 0
rand( ) = 0
rand( ) = 3
rand( ) = 1
rand( ) = 2
Press any key to continue
```



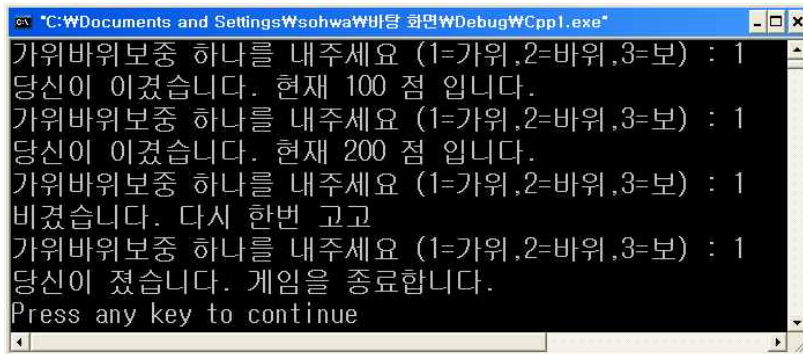
```
rand( ) = 3
rand( ) = 1
rand( ) = 0
rand( ) = 0
rand( ) = 1
rand( ) = 1
rand( ) = 1
rand( ) = 1
rand( ) = 2
rand( ) = 3
Press any key to continue
```



```
rand( ) = 4
rand( ) = 4
rand( ) = 1
rand( ) = 3
rand( ) = 2
rand( ) = 0
rand( ) = 0
rand( ) = 3
rand( ) = 3
rand( ) = 0
Press any key to continue
```

예제 8-63 : srand() 함수를 이용한 간단한 가위바위보 프로그램

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void main ( ) {
    int user, com, jumsu = 0;
    srand( time(NULL) );
    while( 1 ) {
        printf("가위바위보중 하나를 내주세요 (1=가위,2=바위,3=보) : ");
        scanf("%d", user);
        com = rand( ) % 3 + 1;
        if( user == com ) {
            printf("비겼습니다. 다시 한번 고고 \n");
        }
        else if ((user==1 && com==2) || (user==2 && com==3) || (user==3 && com==1)) {
            printf("당신이 졌습니다. 게임을 종료합니다.\n");
            break;
        }
        else {
            jumsu = jumsu + 100;
            printf("당신이 이겼습니다. 현재 %d 점 입니다.\n", jumsu);
        }
    }
}
```



```
"C:\Documents and Settings\Wsohwa\바탕 화면\WDebug\WCpp1.exe"
가위바위보중 하나를 내주세요 (1=가위,2=바위,3=보) : 1
당신이 이겼습니다. 현재 100 점 입니다.
가위바위보중 하나를 내주세요 (1=가위,2=바위,3=보) : 1
당신이 이겼습니다. 현재 200 점 입니다.
가위바위보중 하나를 내주세요 (1=가위,2=바위,3=보) : 1
비겼습니다. 다시 한번 고고
가위바위보중 하나를 내주세요 (1=가위,2=바위,3=보) : 1
당신이 졌습니다. 게임을 종료합니다.
Press any key to continue
```