KG 아이티뱅크

C언어

VISUAL STUDIO

포인터

포인터(pointer)

정의

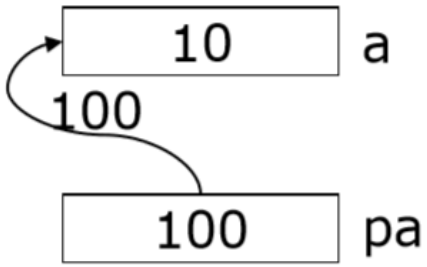
- 특정 주소를 저장하는 변수
- 크기는 4byte (0x0012FF28)

형식

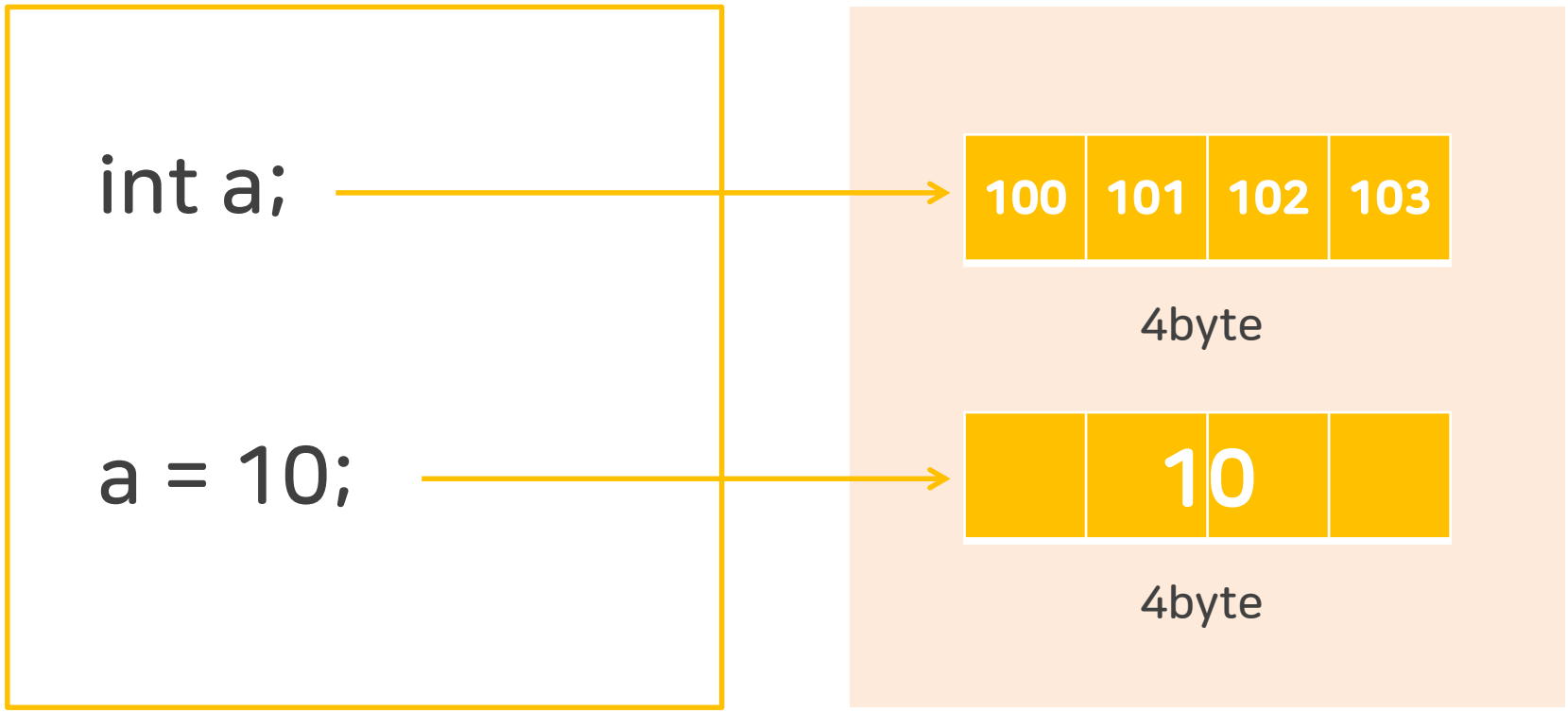
- 자료형 *포인터변수명
- 자료형은 포인터변수에 연결할 변수와 같은 자료형이어야 한다

사용예

```
#include <stdio.h>
main()
{
        int a=10;   //a변수 메모리 할당
        int *pa;    //pa변수 메모리 할당
        pa=&a;      //a의 주소를 pa에 저장
}
```



포인터(pointer)



l-value(left value) : 공간, r-value(right value) : 값
바로 a=10으로 값을 넣어버리면 r-value

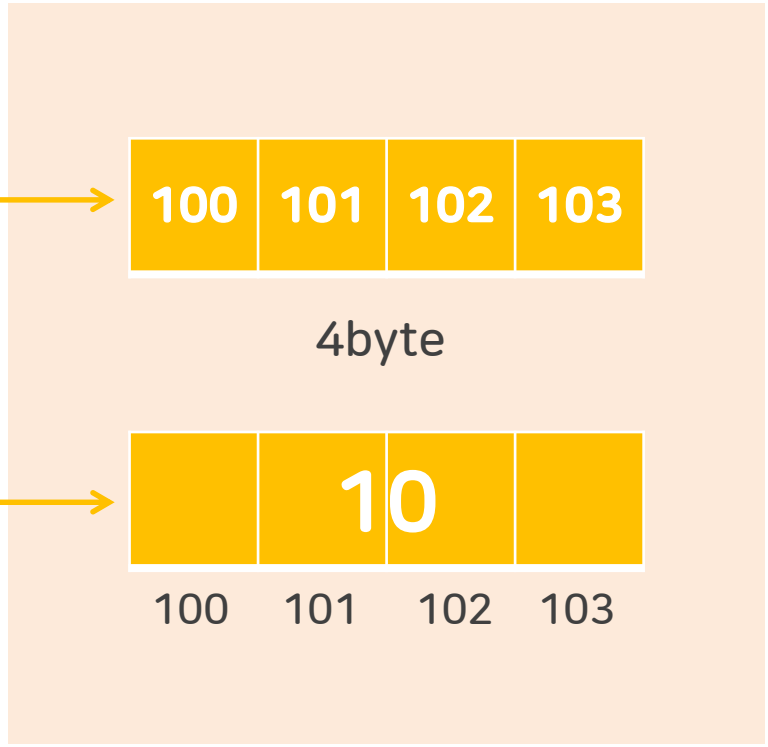
포인터(pointer)

&a => 100번지

a변수의 시작주소

*&a = 10

100번지 주소로 가서 int형
저장공간(4byte)만큼 값을 넣겠다



&a -> int형 저장공간의 시작주소
바로 a = 10을 넣으면 직접참조
*&a = 10은 주소를 통해서 넣는 간접참조

포인터(pointer)

```
int * p;
```

int형 포인터변수 p 선언(4byte)

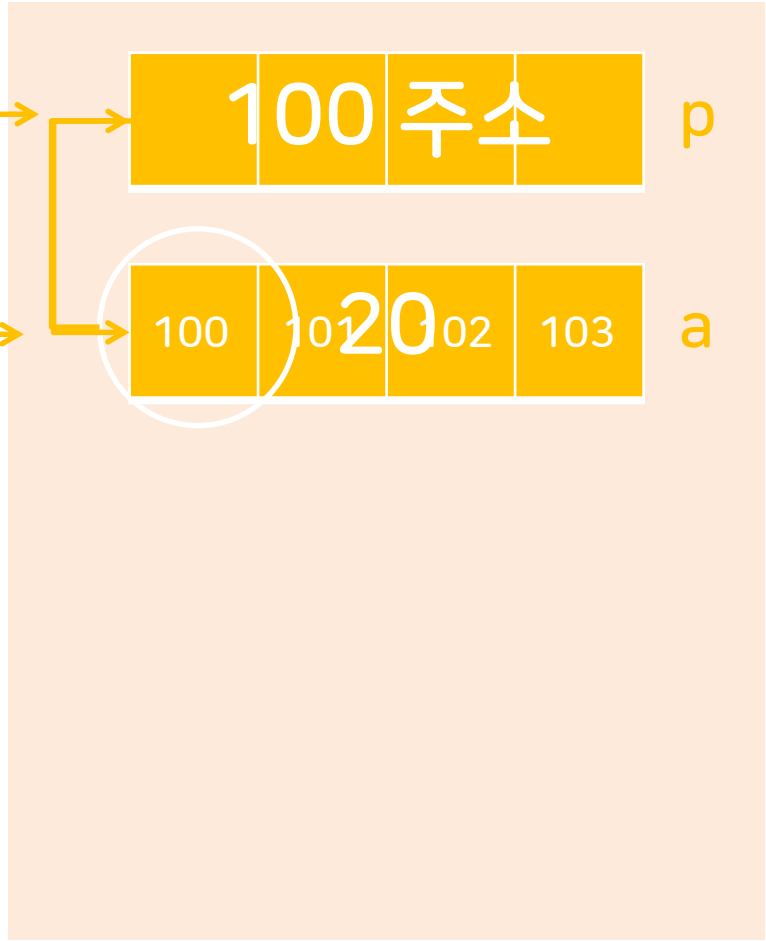
```
int a;
```

```
p = &a;
```

p에 a의 시작주소를 저장

```
*p = 20;
```

p가 저장하고 있는 주소의 값 바꿈



*간접참조연산자를 사용하면 저장된 주소로 이동해서 지시(point)함

포인터(pointer)

<파일이름 : 01.포인터.c>

```
#include<stdio.h>

void main() {
	int num1 = 10, num2;
	int* ptr;

	ptr = &num1;
	num1++;
	num2 = 5;
	ptr = &num2;
	*ptr = *ptr + num1 + num2;
	printf("num1 = %d, num2 = %d, *ptr = %d₩n", num1, num2, *ptr);
}
```

포인터(pointer)

<파일이름 : 02.포인터.c>

```
#include<stdio.h>

void main() {
	char ch1 = 'A', ch2;
	char* ptr;

	ptr = &ch1;
	ch2 = *ptr;
	printf("변수 ch1의 주소는 %p번지 이다.₩n", &ch1);
	printf("변수 ptr의 내용은 %p번지 이다.₩n₩n", ptr);

	printf("변수 ch2의 내용은 '%c' 이다.₩n", ch2);
	printf("변수 *ptr의 내용은 '%c' 이다.₩n₩n", *ptr);

	printf("변수 ch2의 공간의 크기는 %d이다.₩n", sizeof(ch2));
	printf("포인터변수 *ptr이 가르키는 공간의 크기는 %d이다.₩n", sizeof(*ptr));
	printf("포인터변수 ptr의 공간의 크기는 %d이다.₩n", sizeof(ptr));
}
```

포인터(pointer)

<파일이름 : 03.포인터.c>

```
#include<stdio.h>

void main() {
int num = 123, *pnum;
char ch = 'A', *pch;

pnum = &num;
pch = &ch;

printf("pnum-1 = %p번지, pnum = %p번지, pnum+1 = %p번지₩n", pnum - 1, pnum, pnum + 1);
printf("pch-1 = %p번지, pch = %p번지, pch+1 = %p번지₩n", pch - 1, pch, pch + 1);
}
```

포인터(pointer)

<파일이름 : 04.포인터.c>

```
#include<stdio.h>

void main() {
	int num = 10, *pnum = &num;
	char ch = 'A', *pch = &ch;
	float fl = 12.5, *pfl = &fl;
	double dl = 23.24, *pdl = &dl;

	printf("정수형 변수의 크기 : %d₩n", sizeof(num));
	printf("정수형 포인터 변수의 크기 : %d₩n₩n", sizeof(pnum));

	printf("문자형 변수의 크기 : %d₩n", sizeof(ch));
	printf("문자형 포인터 변수의 크기 : %d₩n₩n", sizeof(pch));
}
```

포인터(pointer)

<파일이름 : 04.포인터.c>

```
#include<stdio.h>

void main() {
        int num1 = 5, num2 = 10;
        int *p1 = &num1, *p2 = &num2;

        // 포인터를 활용해서
        num1 과 num2 값을 각각 출력해보세요

        num1 과 num2 의 주소를 각각 출력해보세요

        num1 값에 num2의 값을 더하세요


}
```

포인터(pointer)

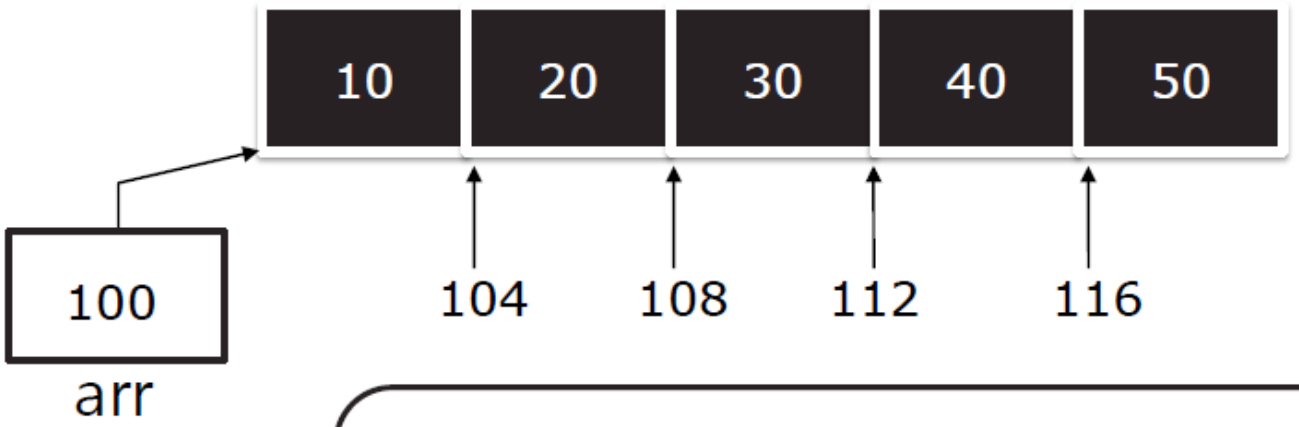
<파일이름 : 05.문제.c>

```
#include<stdio.h>

void main() {
int num1 = 5, num2 = 10;
int *p1 = &num1, *p2 = &num2;

printf("(1) num1 = %d, num2 = %d₩n", num1, num2);
printf("(2) &num1 = %p, &num2 = %p₩n", &num1, &num2);
printf("(3) p1 = %p, p2 = %p₩n", p1,p2);
printf("(4) *p1 = %d, *p2 = %d₩n", *p1,*p2);
printf("(5) *p1 = num1 + num2₩n");
printf("(6) num2 = *p1 + *p2₩n");
printf("(7) num1 = %d, num2 = %d, *p1 = %d, *p2 = %d₩n",num1,num2,*p1,*p2);
}
```

포인터와 배열

- 배열명은 그 배열의 시작 주소를 의미한다. 포인터의 경우 배열의 주소를 받아들이므로 포인터에 배열을 연결하여 사용할 수 있다

- **ex> int arr[5]**



arr[2] → *(arr+(sizeof(int*2))) → *(arr+8)

→ *(100+8) → *(108) → 30 !!!

포인터와 배열

<파일이름 : 06.포인터배열.c>

```
#include<stdio.h>

void main() {
	int a[5] = { 10,20,30,40,50 };
	for (int i = 0; i < 5; i++)
	printf("배열의 주소 %p 배열의 값 %d₩n", a + i, *(a + i));
}
```

포인터와 배열

<파일이름 : 07.포인터배열.c>

```
#include<stdio.h>

void main() {
	int a[5] = { 10,20,30,40,50 };
	int* pa;
	pa = a;
	for (int i = 0; i < 5; i++)
	printf("배열의 주소 %p, 배열의 값%d₩n", pa + i, *(pa + i));
}
```

배열의 이름은 배열의 시작주소를 담고 있으므로 포인터변수에
&를 안 붙이고 이름만 넣어도 주소가 저장

포인터와 배열

<파일이름 : 08.포인터배열.c>

```
#include<stdio.h>

void main() {
	int a[5] = { 10, 20, 30, 40, 50 };
	int *pa;

	pa = a;
	printf("*pa = %d₩n", *pa);
	pa = a+1;
	printf("*pa = %d₩n", *pa);
	pa = a + 2;
	printf("*pa = %d₩n", *pa);

	printf("*pa = %d₩n", *(a+3));
	printf("*pa = %d₩n", *(a+4));
}
```

a[0] = *(a+0) a가 int형 자료이기 때문에 a의 주소에 1을 더하면 int형 한 칸 -> 4가 더해짐

포인터와 배열

<파일이름 : 09.포인터배열.c>

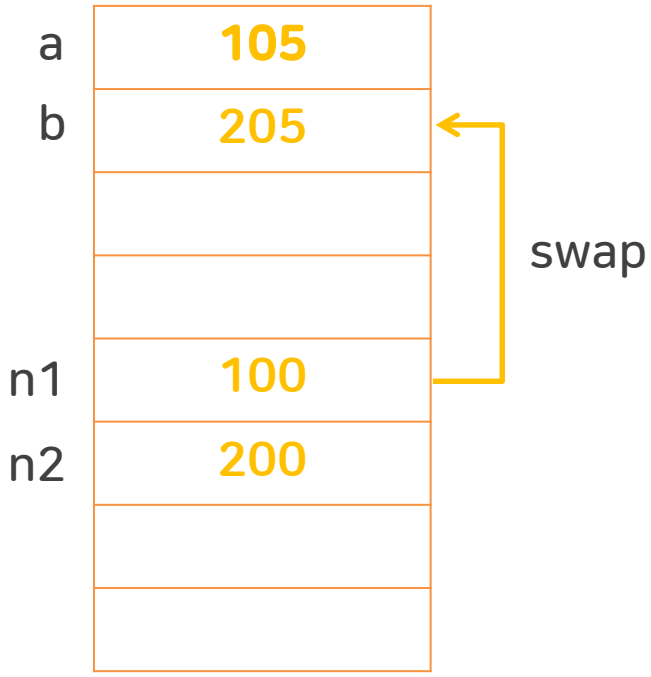
```
#include<stdio.h>

void main() {
int a = 10, b[5] = { 1,2,3,4,5 }, *p;

p = &a;
printf("(1) a= %d, *p=%d₩n", a, *p);

p = b;
printf("(2) a= %d, *p=%d, *(p+1) = %d₩n", a, *p,*(p+1));

p++;
printf("(3) a= %d, *p=%d, *(p+1) = %d₩n", a, *p,*(p+1));
}
```

배열명 b는 주소값을 가지는 상수=> ++등 연산이 불가능
그 주소를 포인터 변수에 넣으면 연산가능

포인터와 배열 – Call by Value

<파일이름 : 10.포인터배열.c>

```
#include <stdio.h>

void func (int a, int b) {
        a += 5;
        b += 5;
        printf("a = %d, b = %d ₩n", a, b);
}

void main () {
        int n1=100, n2=200;

        func(n1, n2);
        printf("n1 = %d, n2 = %d ₩n", n1, n2);
}
```

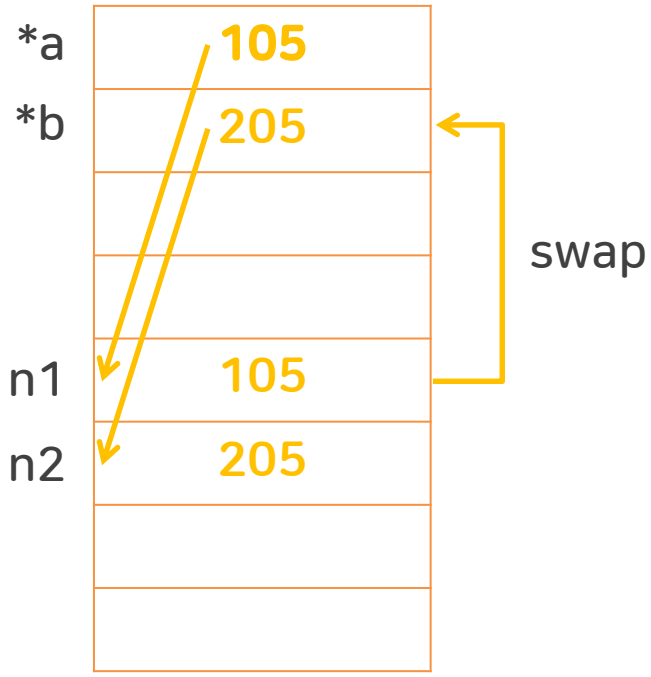


func함수에 직접 인수로 데이터를 넣어줌

포인터와 배열 – Call by Reference

<파일이름 : 11.포인터배열.c>

```
#include <stdio.h>

void func (int* a, int* b) {
        *a += 5;
        *b += 5;
        printf("a = %d, b = %d ₩n", *a, *b);
}

void main () {
        int n1=100, n2=200;

        func(&n1, &n2);
        printf("n1 = %d, n2 = %d ₩n", n1, n2);
}
```



func함수에 n1과 n2의 주소값을 넘겨줌
주소를 받으려면 func함수에서 int* 형태의 포인터 변수 선언

포인터와 배열 – Call by Reference

<파일이름 : 12.포인터배열.c>

```
#include <stdio.h>

void swap(int* x, int* y) {
	int tmp;
	tmp = *x;
	*x = *y;
	*y = tmp;
}

void main() {
	int n1 = 100, n2 = 200;
	printf("교환 전 : n1 = %d, n2 = %d₩n", n1, n2);
	swap(&n1, &n2);
	printf("교환 후 : n1 = %d, n2 = %d₩n", n1, n2);
}
```

swap함수에 n1과 n2의 주소값을 넘겨줌
tmp 에 x의 주소가 가르키는 값을 저장 -> swap

포인터와 함수

<파일이름 : 13.포인터함수.c>

```
#include <stdio.h>

void convertToUppercase(char* s);

void main() {
	char str[] = "characters";

	printf("변환 전의 스트링 : %s₩n", str);
	convertToUppercase(str);
	printf("변환 후의 스트링 : %s₩n", str);
}

void convertToUppercase(char* s)
{
	while (*s != '₩0') {
	if (*s >= 'a' && *s <= 'z')
		*s = *s - 32;
	s++;
	}
}
```

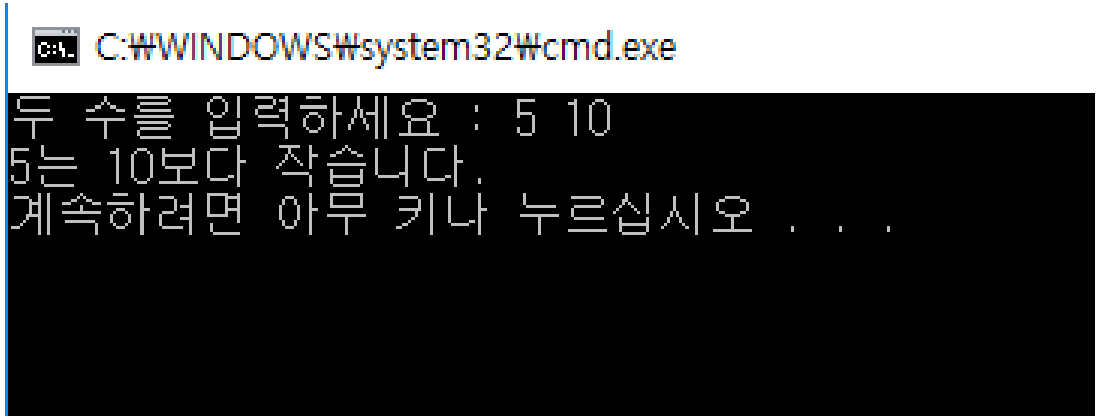
string은 만들 때 부터 배열형태이고 배열의 이름은 배열의 첫 주소를 가지므로 함수에 넘겨줄 때 배열의 이름만 넘겨 줌

문제

<파일이름 : 14.문제.c>

[문제1] 두 수를 입력 받고 포인터변수를 사용하여 큰 수를 출력하시오
<함수>정수형 포인터를 매개변수를 가지고 반환형은 없는 compare선언
<변수> int num1,num2의 주소를 compare로 넘겨줌

<출력>

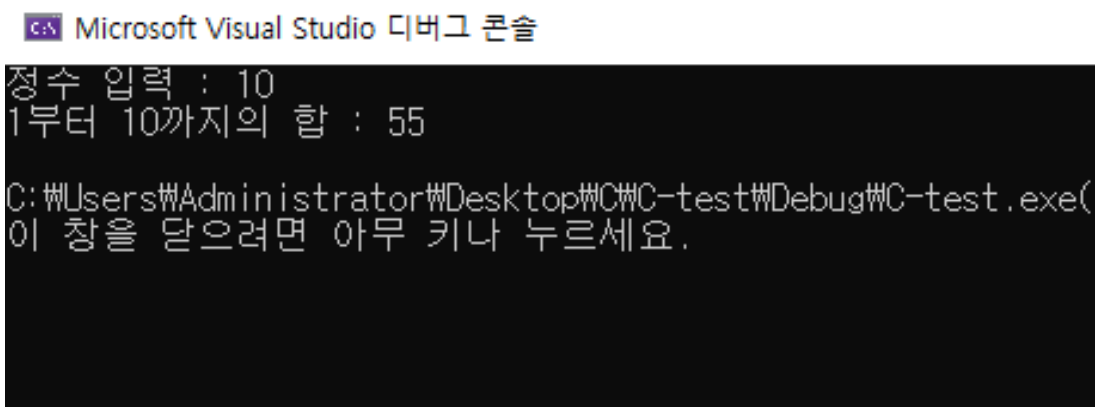


문제

<파일이름 : 15.문제.c>

[문제2] 하나의 숫자를 입력 받아 1 ~ n 까지의 합을 구하는 함수를 만드시오
<함수>정수형 포인터를 매개변수를 가지고 반환형은 없는 funcSum선언
<변수> int num1,num2의 주소를 funcSum으로 넘겨줌

<출력>

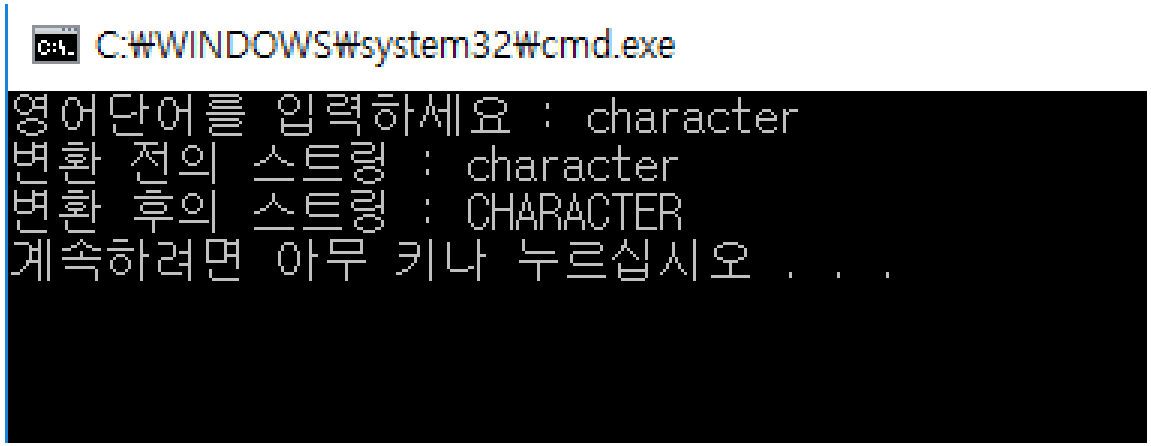


문제

<파일이름 : 16.문제.c>

[문제3] 영어 단어를 입력 받아 영문자를 대문자로 변환하시오.
<함수>정수형 포인터를 매개변수를 가지고 반환형은 없는 convertToUppercase 선언
<변수> char str[]을 선언 -> 입력 받아서 함수에 넘겨 줌

<출력>



다중 포인터

- 다중 포인터 변수는 일반 변수의 주소를 담는 것이 아니라 포인터 변수의 주소를 담을 때 사용한다.

- 사용예

```
#include <stdio.h>
main()
{
        int a=10;           //a변수 메모리 할당
        int *pa=&a;          //일반 변수(int) a의 주소를 포인터 변수pa에 저장
        int **ppa=&pa;       // 포인터 변수 pa의 주소를 다중 포인터변수 ppa에 저장
        int ***pppa=&ppa;
}
```

다중 포인터

<파일이름 : 17.다중포인터.c>

```
#include<stdio.h>

void changeNum(int** p) {
	**p = 10;
}

void main() {
	int num = 5;
	int* p = &num;
	printf("바꾸기 전 : %d₩n", *p);
	changeNum(&p);
	printf("바꾼 후 : %d₩n", *p);
}
```

포인터 변수 p에 num의 주소를 저장 p를 함수에 넘겨주면 num의 주소를 저장하고 있는 p의 주소를 저장하기 위해서 int** p 이렇게 두 번 지시(point)할 수 있는 변수를 인수로 받아서 사용

다중 포인터

<파일이름 : 18.다중포인터.c>

```
#include <stdio.h>

void Swap(int** pp_n1, int** pp_n2) {
		// 이중포인터를 이용해 스왑을 구현
}

void main() {
	int n1 = 10, n2 = 20;
	int* p_n1, *p_n2;

	p_n1 = &n1;
	p_n2 = &n2;

	Swap(&p_n1, &p_n2);

	printf("n1 : %d, n2 : %d ₩n", n1, n2);
}
```