포인터

1주차

- 주소
- 포인터 변수
- 포인터변수의 자료유형 변환과 연산

주소 (address)

- 메모리 1 byte 마다 연속적으로 매겨진 번호(0 부터)

- 16진수로 표현
  - 4 byte이므로 16진수 8자리
  - 0x00000000

- “변수” 외에 “주소”로도 저장장소를 참조할 수 있음

주소연산자 (&)

- 변수의 전위연산자로서 그 변수의 주소값을 반환
- 출력할 때는 변환명세 %p 사용

[코드]

```
int  age = 10;
printf(  "%d, %#p" , age, &age );
```

[출력결과]

10, 0x003DFD78

함수 scanf() : 변환문자에 대응되는 개수의 주소를 줘야 함.
일반 변수에 대해, '&변수이름' 으로 사용.

주소연산자 예제

int, double, char 자료형의 변수에 초기값을 주어 선언하고
이 변수들의 값과 주소값을 출력하시오.

```
#include <stdio.h>

void main(void)
{
    int n = 10;
    double d = 10.5;
    char ch = '1';

    printf( "n=%d &n=%p\n", n, &n );
    printf( "d=%g &d=%#p\n", d, &d );
    printf( "ch=%c &ch=0x%p\n", ch, &ch );
}
```

포인터 변수

- 포인터 변수란
  - 주소값을 저장하는 변수
  - 모든 자료형에 대하여 포인터 변수를 만들 수 있음
  - 선언 시 변수명 앞에 * 를 붙임.

```
int  *ptr;
int  *ptr1, *ptr2, *ptr3;
```

  - 값이 없는 포인터 NULL (=0)

```
int  *ptr = NULL;
```

포인터 변수에 다른 변수의 주소값 저장하기

- 주소값을 저장하려는 변수와 포인터 변수의 **자료형이 반드시 일치**해야 함

```
double  pi = 3.14;
double *pPi = &pi;
```

```
char  ch =  '@' ;
int  *pCh = &ch;
```

  - 포인터 변수 pPi는 변수 pi를 가리킨다 (=참조한다)

포인터변수 예제

```
#include <stdio.h>

void main(void)
{
    int n = 10;
    double d = 10.5;
    char ch = '1';
    int *pn;
    double *pd;
    char *pch;

    pn = &n;
    pd = &d;
    pch = &ch;

    printf( "n=%d &n=%p pn=%p\n", n, &n, pn );
    printf( "d=%g &d=%#p pd=%#p\n", d, &d, pd );
    printf( "ch=%c &ch=0x%p pch=0x%#p\n", ch, &ch, pch );
}
```

결과에 유의!
pn = &n;
한 후에는
&n 과 pn 이
같은 의미, 같은 값임을 확인!

간접 연산자 * 를 이용한 간접 참조

- 포인터변수의 전위연산자
- 포인터가 가리키는 변수 자체를 반환

```
int  i = 10;
int  *p;
p = &i;
*p = *p + 1;
```

이로 인해 다음이 성립.

p == &i
*p == i

위와 같을 때 아래가 성립함
(1) 변수 p에는 변수i의 주소값이 있음
(2) i 와 *p 는 동일한 int 변수
(3) 변수 i의 값이 11로 바뀜

저장장소 참조방법

실제 저장 장소 내용

도식화된 표현

프로그램

```
int  num = 10;
int  *pNum;
*pNum = &num;
```

* 연산자 사용 예 (dereference.c)

```
#include <stdio.h>

int main(void)
{
    int data = 100;
    char ch = 'A';
    int *ptrint = &data;
    char *ptrchar = &ch;
    printf("간접참조 출력: %d %c₩n", *ptrint, *ptrchar);
    *ptrint = 200;
    *ptrchar = 'B';
    printf("직접참조 출력: %d %c₩n", data, ch);

    return 0;
}
```

간접참조에 의한 내용 수정

[결과]
간접참조 출력: 100 A
직접참조 출력: 200 B

주소 연산

- 포인터 변수의 연산
  - 덧셈, 뺄셈

- 포인터가 가리키는 변수 타입의 크기를 반영함
  - int *pi; 에서 pi에 저장된 주소값이 100일 때
  - (pi+1)은 1010이 아니라 주소값 104
  - 즉 (pi+1)은 pi가 가리키는 다음 int형의 주소를 의미

주소 연산 예 (calcptr.c)

```
#include <stdio.h>

int main(void) {
        char *pc = (char *)100;
        int *pi = (int *)100;
        double *pd = (double *)100;
        pd = 100;

        printf("%u  %u  %u₩n", (int)(pc - 1), (int)pc, (int)(pc + 1));
        printf("%u  %u  %u₩n", (int)(pi - 1), (int)pi, (int)(pi + 1));
        printf("%u  %u  %u₩n", (int)(pd - 1), (int)pd, (int)(pd + 1));

        return 0;
}
```

가능은 하지만 이렇게 사용 하지 않음! 이유는?

[결과]

```
99  100  101
96  100  104
92  100  108
```

포인터를 이용하여 두 수의 값을 교환하는 프로그램

```
// file: swap.c
#include <stdio.h>

int main(void) {
    int m = 100, n = 200, dummy;
    printf("%d %d₩n", m, n);

    int *p = &m;
    dummy = *p;
    *p = n;
    p = &n;
    *p = dummy;

    printf("%d %d₩n", m, n);

    return 0;
}
```

변수 m과 n을 사용하지 않고 두 변수를 서로 교환

[결과]
100 200
200 100

내부 저장 표현
(little endian <-> big endian)

```
// file: ptrtypecast.c
#include <stdio.h>

int main(void) {
    int value = 0x61626364;
    int *pi = &value;
    char *pc = (char *)&value;

    printf("변수명    저장값       주소값₩n");
    printf("--------------------------₩n");
    printf(" value    %0#x  %p₩n", value, pi); //정수 출력
    for (int i = 0; i < 4; i++) {
        char ch = *(pc + i);
        printf("*(pc+%d)  %0#6x  %2c  %p₩n", i, ch, ch, pc + i);
    }
    return 0;
}
```

명시적 형변환

문자 포인터(1 byte씩 움직임)로 문자 출력

[결과]

```
변수명    저장값       주소값
--------------------------
 value    0x61626364  007DF7A0
*(pc+0)   0x0064   d  007DF7A0
*(pc+1)   0x0063   c  007DF7A1
*(pc+2)   0x0062   b  007DF7A2
*(pc+3)   0x0061   a  007DF7A3
```