포인터의 이해

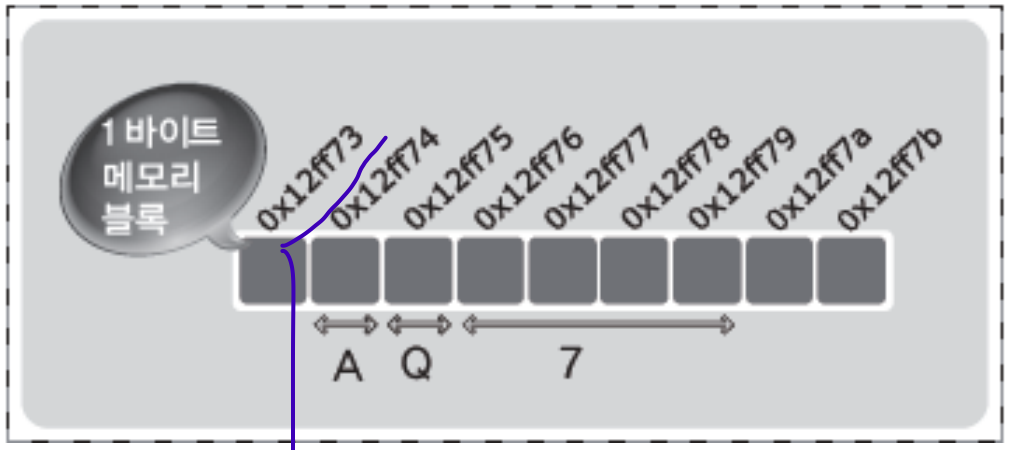
- 포인터 변수와 포인터 형 -

성공회대학교 IT융합자율학부
소프트웨어공학전공
홍 성 준

포인터란 무엇인가?

◎ 메모리 상에서 변수의 할당

```
int main(void)
{
    char ch1='A', ch2='Q';
    int num=7;
    . . . .
}
```

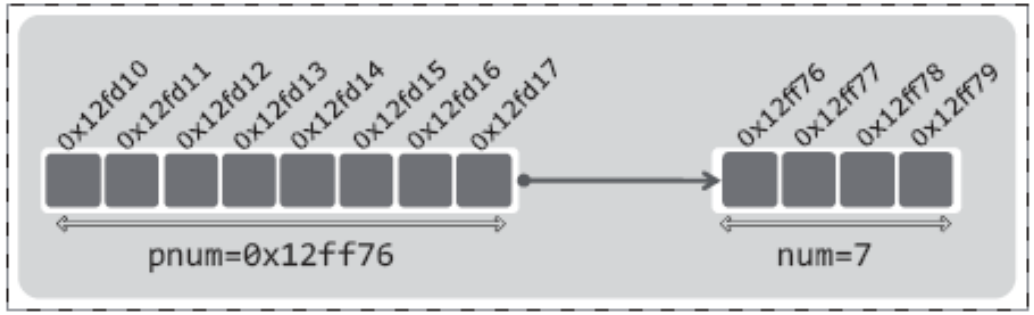


- 메모리 블록(1바이트)을 단위로 하나의 주소 값이 할당되며 주소값은 1씩 증가함

포인터란 무엇인가?

◎ 포인터 변수

- 정수 형태의 메모리 주소 값을 저장하기 위한 변수



포인터와 변수의 참조 관계

◎ 포인터 변수의 선언의 예

```
int main(void)
{
      int num=7;
      int * pnum;
      pnum = &num;
      . . . .
}
```

pnum	포인터 변수의 이름
int *	int형 변수의 주소 값을 저장하는 포인터 변수의 선언

- int형 변수 num을 선언하고 7로 초기화 한 후, 이 변수의 주소 값 저장을 위한 포인터 변수 pnum을 선언하여 pnum 변수에 변수 num의 주소 값을 저장하자.
- 포인터 변수 pnum이 변수 num을 가리킨다.

포인터 선언하기

◎ 포인터 변수의 선언

- 포인터 변수에 저장되는 주소 값은 정수로 형태가 동일하지만, 가리키고자 하는 변수의 자료 형에 따라 선언 방법이 다름

```
type * ptr;
```

type형 변수의 주소 값을 저장하는 포인터 변수 ptr의 선언

```
int * pnum1;
```

int * 는 int형 변수를 가리키는 pnum1의 선언을 의미함

```
double * pnum2;
```

double * 는 double형 변수를 가리키는 pnum2의 선언을 의미함

```
unsigned int * pnum3;
```

unsigned int * 는 unsigned int형 변수를 가리키는 pnum3의 선언을 의미함

포인터 선언하기

◎ 포인터 선언에서 * 의 위치

- int * ptr;
- int* ptr;
- int *ptr;

◎ 포인터 변수는 일반 변수와 함께 선언 가능

- int num, *pnum;
- int num;

  int *pnum;

◎ 포인터 변수의 크기

- 포인터 변수 크기는 시스템에 따라 4바이트 혹은 8바이트를 사용 (주소 값의 크기와 포인터 변수의 크기가 동일하게)

포인터의 형

◎ 포인터의 형(type)

- 포인터 변수의 선언 및 구분에 사용하는 int *, char *, double * 등을 포인터 형(type)이라고 함
- 가리키려는 변수의 데이터 형에 따라 포인터의 형도 동일하게 선언해줘야 함

```
int *              int형 포인터
int * pnum1;       int형 포인터 변수 pnum1

double *           double형 포인터
double * pnum2;    double형 포인터 변수 pnum2
```

포인터 연산자

◎ & 연산자

- 피연산자(변수)의 주소를 반환하는 단항 연산자

```
int main(void)
{
     int num = 5;
     int * pnum = &num;
     . . . .
}
```

```
int main(void)
{
     int num1 = 5;
     double * pnum1 = &num1;   // 일치하지 않음!

     double num2 = 5;
     int * pnum2 = &num2;   // 일치하지 않음!
     . . . .
}
```

- 포인터 변수의 형과 대입하려는 포인터 형이 일치하지 않으면 경고 메시지 발생

포인터 연산자

◎ * 연산자

- 포인터가 가리키는 메모리 공간에 접근하는 단항 연산자

```
int main(void)
{
     int num=10;
     int * pnum=&num;
     *pnum=20;
     printf("%d", *pnum);
     . . . .
}
```

  - *pnum 은 num 을 의미하여 num을 놓을 자리에 *pnum을 놓을 수 있음

포인터 연산자

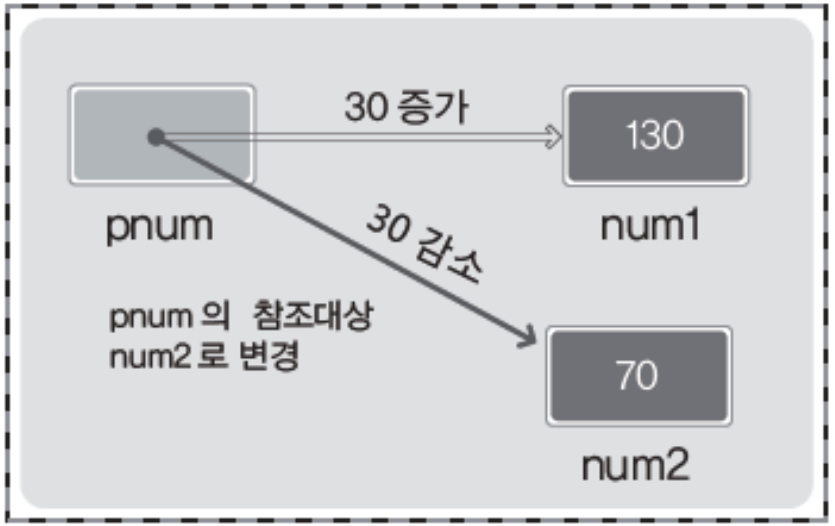
◎ PointerOperation.c

```
int main(void)
{
    int num1=100, num2=100;
    int * pnum;

    pnum=&num1;     // 포인터 pnum이 num1을 가리킴
    (*pnum)+=30;    // num1+=30; 과 동일

    pnum=&num2;    // 포인터 pnum이 num2를 가리킴
    (*pnum)-=30;    // num2-=30; 과 동일

    printf("num1:%d, num2:%d \n", num1, num2);
    return 0;
}
```




```
num1:130, num2:70
```

포인터 연산자

◎ 다양한 포인터 형을 사용하는 이유

- 포인터 형은 메모리 공간을 참조하는 기준
- * 연산을 통해 메모리의 접근 기준을 마련하기 위해서 다양한 포인터 형을 사용
  - int 형 포인터 변수로 메모리에 접근 시 4바이트 메모리 공간에 있는 데이터를 정수로 읽어 옴
  - double 형 포인터 변수로 메모리에 접근 시 8바이트 메모리 공간에 있는 데이터를 실수로 읽어 옴

```
int main(void)
{
     double num=3.14;
     int * pnum=&num;
     printf("%d", *pnum);
     . . . .
}
```

1374389535

포인터의 선언과 초기화

◎ 포인터의 초기화

```
int main(void)
{
     int * ptr;
     *ptr=200;
     . . . .
}
```

```
int main(void)
{
     int * ptr=125;
     *ptr=10;
     . . . .
}
```

포인터의 선언과 초기화

◎ 포인터의 초기화

- NULL : 널 포인터 (null pointer)
  - 널 포인터 NULL은 숫자 0을 의미하며 0번지를 뜻하는 것이 아닌 아무것도 가리키기 않는다는 의미
  - 잘못된 포인터 연산을 막기 위해 포인터 변수 선언 시 널 포인터를 이용하여 초기화

```
int main(void)
{
     int * ptr1=0;
     int * ptr2=NULL;
     . . . .
}
```