Pointers

adopted from KNK C Programming : A Modern Approach

Pointers as Arguments

- In Chapter 9, we tried—and failed—to write a `decompose` function that could modify its arguments.
  9장에서 decompose 함수가 실수 부분과 소수 부분을 분리하는 것을 실패하였음

- By passing a *pointer* to a variable instead of the *value* of the variable, `decompose` can be fixed.
  변수의 값 대신 변수에 대한 포인터를 전달하는 것으로 문제를 해결할 수 있음

Pointers as Arguments

- New definition of `decompose`: 새로운 decompose 함수의 정의

```
void decompose(double x, long *int_part,
               double *frac_part)
{
  *int_part = (long) x;
  *frac_part = x - *int_part;
}
```

- Possible prototypes for `decompose`: 이 함수의 프로토타입

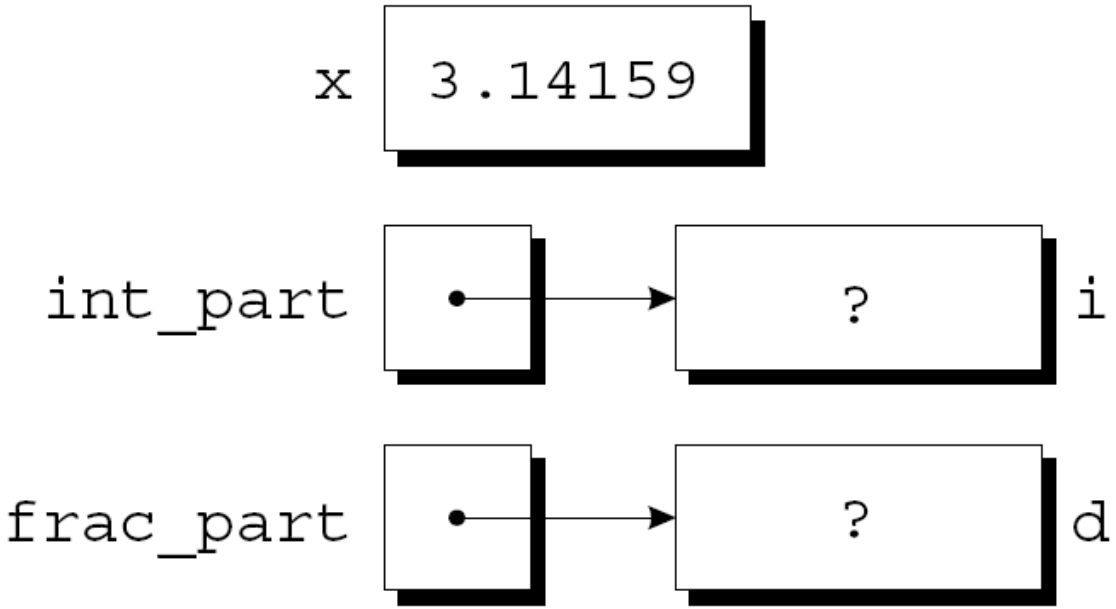
```
void decompose(double x, long *int_part,
               double *frac_part);
void decompose(double, long *, double *);
```

Pointers as Arguments

- A call of `decompose`: 함수 호출 방법

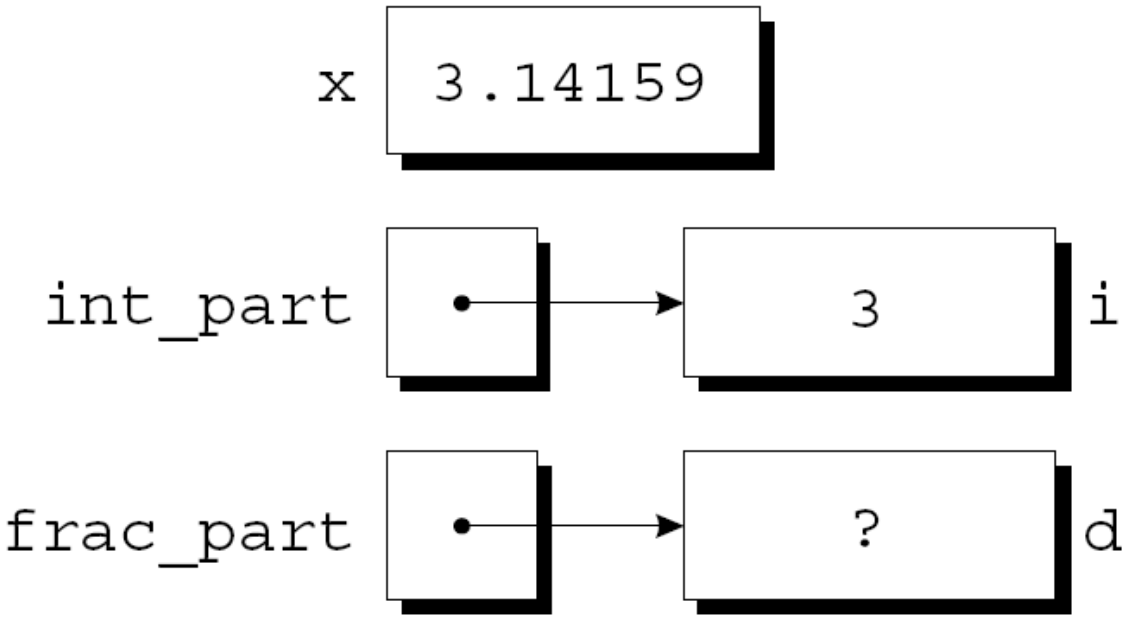
```
decompose(3.14159, &i, &d);
```

- As a result of the call, `int_part` points to `i` and `frac_part` points to `d`:
  호출의 결과로 int_part는 i를 가리키고 frac_part는 d를 가리킴



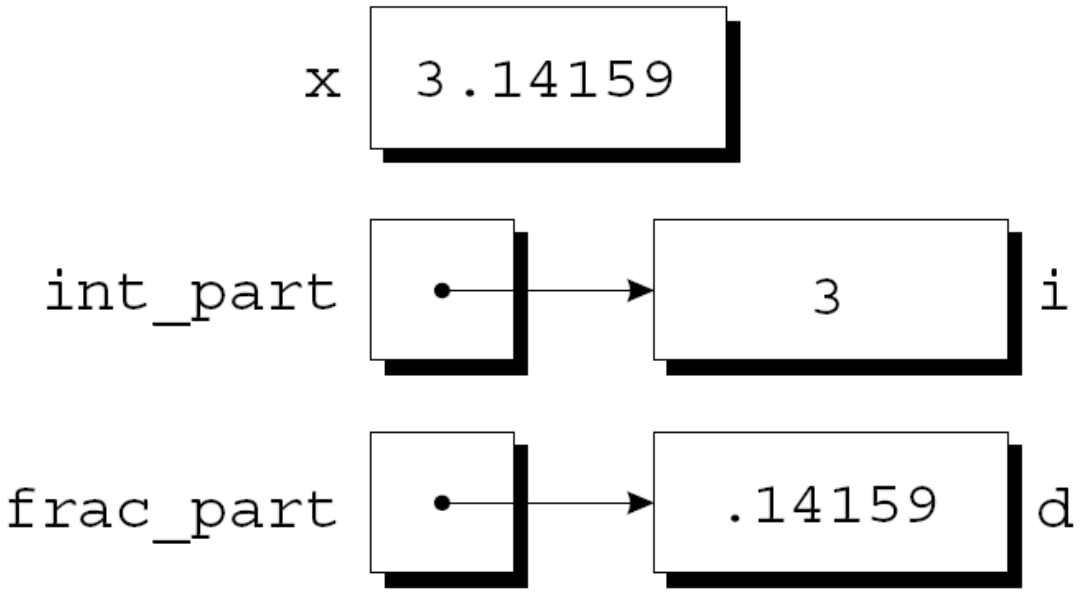
Pointers as Arguments

- The first assignment in the body of `decompose` converts the value of `x` to type `long` and stores it in the object pointed to by `int_part`:
  decompse 함수의 첫 할당문은 x의 값을 long 타입으로 변경한 후 int_part가 가리키는 객체에 저장함



Pointers as Arguments

- The second assignment stores `x - *int_part` into the object that `frac_part` points to:
  두 번째 할당문은 x - *int_part 의 계산 결과를 frac_part가 가리키는 객체에 저장함



Program: Finding the Largest and Smallest Elements in an Array

- The `max_min.c` program uses a function named `max_min` to find the largest and smallest elements in an array.
  max_min.c이라는 프로그램은 max_min 함수를 사용하여 배열에서 가장 큰 수와 작은 수를 찾음
- Prototype for `max_min`: 프로토타입은 다음과 같음

```
void max_min(int a[], int n, int *max, int *min);
```

- Example call of `max_min`: 호출 예제

```
max_min(b, N, &big, &small);
```

- When `max_min` finds the largest element in `b`, it stores the value in `big` by assigning it to `*max`. 이 함수가 배열 b에서 가장 큰 요소를 찾으면 *max를 통해서 big에 할당
- `max_min` stores the smallest element of `b` in `small` by assigning it to `*min`. 이 함수가 배열 b에서 가장 작은 요소를 찾으면 *min을 통해서 small에 할당

Program: Finding the Largest and Smallest Elements in an Array

- `max_min.c` will read 10 numbers into an array, pass it to the `max_min` function, and print the results:
  사용자로부터 10개의 수를 받아 들이고 max_min함수에 전달함. 이후 결과를 출력

```
Enter 10 numbers: 34 82 49 102 7 94 23 11 50 31
Largest: 102
Smallest: 7
```

maxmin.c

```
/* Finds the largest and smallest elements in an array */

#include <stdio.h>

#define N 10

void max_min(int a[], int n, int *max, int *min);

int main(void)
{
  int b[N], i, big, small;

  printf("Enter %d numbers: ", N);
  for (i = 0; i < N; i++)
    scanf("%d", &b[i]);
```

```
  max_min(b, N, &big, &small);

  printf("Largest: %d\n", big);
  printf("Smallest: %d\n", small);

  return 0;
}

void max_min(int a[], int n, int *max, int *min)
{
  int i;

  *max = *min = a[0];
  for (i = 1; i < n; i++) {
    if (a[i] > *max)
      *max = a[i];
    else if (a[i] < *min)
      *min = a[i];
  }
}
```

Pointers and Arrays

adopted from KNK C Programming : A Modern Approach

Program: Reversing a Series of Numbers (Revisited)

- The `reverse.c` program of Chapter 8 reads 10 numbers, then writes the numbers in reverse order.
  8장에서 reverse.c프로그램은 숫자를 역순으로 출력함
- The original program stores the numbers in an array, with subscripting used to access elements of the array.
  원 프로그램은 수를 배열에 저장하였고, 역순으로 배열 첨자를 순회하였음
- `reverse3.c` is a new version of the program in which subscripting has been replaced with pointer arithmetic.
  reverse3.c는 첨자 대신 포인터 연산을 활용함

reverse3.c

```
/* Reverses a series of numbers (pointer version) */

#include <stdio.h>

#define N 10

int main(void)
{
  int a[N], *p;

  printf("Enter %d numbers: ", N);
  for (p = a; p < a + N; p++)
    scanf("%d", p);

  printf("In reverse order:");
  for (p = a + N - 1; p >= a; p--)
    printf(" %d", *p);
  printf("\n");

  return 0;
}
```