3주차 - 함수 및 포인터

다음을 수행해야 합니다:

1. 실습 문제 - 실습 문제는 실습 세션 중에 수행하세요. 자동화된 평가 시스템은 프로그램 입력/출력에 대한 정확한 문자열 매칭을 사용하는 테스트 사례를 기반으로 하므로 실습 문제를 풀 때 프로그램 입력/출력에 대한 질문 요구 사항을 정확히 따르시기 바랍니다.
2. 실습 과제 문제 - 과제 문제를 풀고 채점을 위해 온라인 자동 프로그래밍 평가 시스템(APAS)에 코드를 제출하세요.

튜터 이 실습 튜토리얼 세션에서는 실습실에서 각 문제에 대한 해결책에 대해 토론해 주세요. 각 문제에

실험실 질문

질문 1-3

그림 1의 프로그램 템플릿을 사용하여 다음 세 가지 질문에서 함수를 테스트할 수 있습니다. 이 프로그램에는 사용자가 다음 함수를 테스트할 수 있도록 switch 문이 포함된 **main**() 함수가 포함되어 있습니다. 각 함수에 대한 코드를 작성하고 제안된 테스트 케이스를 사용하여 코드가 올바른지 테스트하세요.

```
#include <stdio.h>
/* 함수 프로토타입 */ int
numDigits1(int num);
int digitPos1(int num, int digit);
int square1(int num);
void numDigits2(int num, int *result);
void digitPos2(int num, int digit, int *result);
void square2(int num, int *result);

int main()
{
   int 선택;
   int number, digit, result=0;
   do { {
      printf("\n다음 함수를 반복적으로 수행합니다:\n"); printf("1:
      numDigits1()\n");
      printf("2: numDigits2()\n");
      printf("3: digitPos1()\n");
      printf("4: digitPos2()\n");
      printf("5: square1()\n");
      printf("6: square2()\n");
      printf("7: 종료\n"); printf("선
      택 항목 입력: "); scanf("%d",
      &choice);

      스위치 (선택) { 케이스
         1:
            printf("숫자 입력: \n"); scanf("%d",
            &number);
            printf("numDigits1(): d\n", numDigits1(number));
            break;
         사례 2:
            printf("숫자 입력: \n"); scanf("%d",
            &number); numDigits2(number, &result);
            printf("numDigits2(): d\n", result);
            break;
```

```
            사례 3:
                printf("숫자를 입력하세요: \n");
                scanf("%d", &number); printf("숫자
                를 입력하세요: \n"); scanf("%d",
                &digit);
                printf("digitPos1(): d\n", digitPos1(숫자, 숫자));
                break;
            사례 4:
                printf("숫자 입력: \n"); scanf("%d",
                &number); printf("숫자 입력: \n");
                scanf("%d", &digit);
                digitPos2(number, digit, &result);
                printf("digitPos2(): %d\n", result);
                break;
            사례 5:
                printf("숫자 입력: \n"); scanf("%d",
                &number);
                printf("square1(): d\n", square1(number));
                break;
            사례 6:
                printf("숫자 입력: \n"); scanf("%d",
                &number); square2(number, &result);
                printf("square2(): d\n", result);
                break;
            default: printf("프로그램 종료 중.....\n");
                break;
        }
    } while (choice < 7);
    return 0;
}
/* 여기에 함수 코드 추가 */
    int numDigits1(int num)
    {
        int count = 0;

        do {
            count++;
            숫자 = 숫자/10;
        } while (num > 0); 반
        환 카운트;
    }
    void numDigits2(int num, int *result)
    {
        *결과=0;
        /* 여기에 프로그램 코드를 작성하세요 */
    }
    int digitPos1(int num, int digit)
    {
        /* 여기에 프로그램 코드를 작성하세요 */
    }
    void digitPos2(int num, int digit, int *result)
    {
        int pos=0;
        *result=0;
        do {
            pos++;
            if (num%10 == digit){
                *result = pos;
                break;
            }
            숫자 = 숫자/10;
        } 동안 (num > 0);
```

```
}
int square1(int num)
{
    /* 여기에 프로그램 코드를 작성하세요 */
}
void square2(int num, int *result)
{
    /* 여기에 프로그램 코드를 작성하세요 */
}
```

그림 1

1. **(numDigits)** 음수가 아닌 정수의 자릿수를 세는 함수를 작성합니다. 예를 들어 1234는 자릿수가 4자리입니다. **numDigits1()** 함수는 결과를 반환합니다. 함수 프로토타입은 아래와 같습니다:

```
int numDigits1(int num);
```

포인터 매개변수인 *결과값을* 통해 결과를 전달하는 또 다른 함수 **numDigits2()**를 작성합니다. 함수 프로토타입은 아래와 같습니다:

```
void numDigits2(int num, int *result);
```

몇 가지 샘플 입력 및 출력 세션이 아래에 나와 있습니다:

(1) 테스트 사례 1:
번호를 입력합니다:
1
numDigits1(): 1
numDigits2(): 1

(2) 테스트 사례 2:
번호를 입력합니다:
13579
numDigits1(): 5
numDigits2(): 5

별도의 프로그램 테스트용: 다음 샘플 프로그램 템플릿은 아래에 나와 있습니다:

```
#include <stdio.h>
int numDigits1(int num);
void numDigits2(int num, int *result);
int main()
{
    int 숫자, 결과=0;

    printf("숫자 입력: \n"); scanf("%d",
    &number);
    printf("numDigits1(): %d\n", numDigits1(number));
    numDigits2(number, &result);
    printf("numDigits2(): d\n", result);
    0을 반환합니다;
}
int numDigits1(int num)
{
    /* 여기에 프로그램 코드를 작성하세요 */
}
void numDigits2(int num, int *result)
{
    /* 여기에 프로그램 코드를 작성하세요 */
}
```

2. **(digitPos)** 지정된 숫자의 첫 번째 출현 위치를 양수로 반환하는 함수 **digitPos1()을** 작성합니다. 숫자의 위치는 오른쪽부터 카운트되며 1부터 시작합니다. 필요한 숫자가 숫자에 없으면 함수는 0을 반환해야 합니다. 예를 들어 digitPos1(12315, 1)은 2를 반환하고 digitPos1(12, 3)은 0을 반환합니다. 함수 프로토타입은 아래에 나와 있습니다:

```
int digitPos1(int num, int digit);
```

포인터 매개변수인 *결과값을* 통해 결과를 전달하는 다른 함수 **digitPos2()를** 작성합니다. 예를 들어, num = 12315이고 digit = 1이면 *result = 2, num = 12이고 digit = 3이면 *result.
= 0. 함수 프로토타입은 아래와 같습니다:

```
void digitPos2(int num, int digit, int *result);
```

몇 가지 샘플 입력 및 출력 세션이 아래에 나와 있습니다:

(1) 테스트 사례 1:
```
번호를 입력합니다:
234567
숫자를 입력합니다:
6
digitPos1(): 2
digitPos2(): 2
```

(2) 테스트 사례 2:
```
번호를 입력합니다:
234567
숫자를 입력합니다:
8
digitPos1(): 0
digitPos2(): 0
```

별도의 프로그램 테스트용: 다음 샘플 프로그램 템플릿은 아래에 나와 있습니다:

```
#include <stdio.h>
int digitPos1(int num, int digit);
void digitPos2(int num, int digit, int *result);
int main()
{
    int 숫자, 숫자, 결과=0;

    printf("숫자를 입력하세요: \n");
    scanf("%d", &number); printf("숫자
    를 입력하세요: \n"); scanf("%d",
    &digit);
    printf("digitPos1(): %d\n", digitPos1(number, digit));
    digitPos2(number, digit, &result);
    printf("digitPos2(): d\n", result);
    0을 반환합니다;
}
int digitPos1(int num, int digit)
{
    /* 여기에 프로그램 코드를 작성하세요 */
}
void digitPos2(int num, int digit, int *result)
{
    /* 여기에 프로그램 코드를 작성하세요 */
```

```
}
```

3. **(**제곱**)** 아래 예제와 같이 1로 시작하는 홀수 정수의 합을 계산하여 양의 정수 *num*의 제곱을 반환하는 함수 **square1()**을 작성합니다. 이 함수의

결과가 호출 함수에 반환됩니다. 예를 들어 *num* = 4이면 $4^2 = 1 + 3 + 5 + 7 = 16$ 이 반환되고, *num* = 5이면 $5^2 = 1 + 3 + 5 + 7 + 9 = 25$가 반환됩니다. 함수 프로토타입입니다:

```
int square1(int num);
```

결과를 포인터 매개변수 *result*로 전달하는 다른 함수 **square2()**를 작성합니다. 예를 들어 *num* = 4이면 *result = $4^2 = 1 + 3 + 5 + 7 = 16$이고, *num* = 5이면 *result = $5^2 = 1 + 3 + 5 + 7 + 9 = 25$. 함수 프로토타입은 다음과 같습니다:

```
void square2(int num, int *result);
```

몇 가지 샘플 입력 및 출력 세션이 아래에 나와 있습니다:

(1) 테스트 사례 1:
```
번호를 입력합니다:
4
square1(): 16
square2(): 16
```

(2) 테스트 사례 2:
```
번호를 입력합니다:
0
square1(): 0
square2(): 0
```

별도의 프로그램 테스트용: 다음 샘플 프로그램 템플릿은 아래에 나와 있습니다:

```
#include <stdio.h>
int square1(int num);
void square2(int num, int *result);
int main()
{
   int 숫자, 결과=0;

   printf("숫자 입력: \n"); scanf("%d",
   &number);
   printf("square1(): d\n", square1(number));
   square2(number, &result);
   printf("square2(): d\n", 결과);
   0을 반환합니다;
}
int square1(int num)
{
   /* 여기에 프로그램 코드를 작성하세요 */
}
void square2(int num, int *result)
{
   /* 여기에 프로그램 코드를 작성하세요 */
}
```

4. **(calDistance)** 평면에 있는 두 점, 즉 (x1, y1)과 (x2, y2)의 좌표를 나타내는 10진수 값 4개를 받아 점 사이의 거리를 계산하여 표시하는 C 프로그램을 작성하세요:

$$\text{거리} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

프로그램은 함수를 사용하여 구현해야 합니다. 거리를 계산하는 함수의 두 가지 버전을 제공하십시오. (a) 하나는 매개 변수를 전달하는 데만 값으로 호출을 사용하고, (b) 다른 하나는 호출 함수에 결과를 전달하는 데 참조로 호출을 사용합니다.

함수 프로토타입은 아래와 같습니다:

```
void inputXY(double *x1, double *y1, double *x2, double *y2);
void outputResult(double dist);
double calDistance1(double x1, double y1, double x2, double y2);
void calDistance2(double x1, double y1, double x2, double y2),
                  double *dist);
```

C 프로그램을 작성하여 기능을 테스트합니다.

몇 가지 샘플 입력 및 출력 세션이 아래에 나와 있습니다:

(1) 테스트 사례 1:

```
x1 y1 x2 y2를 입력합니다:
1 1 5 5
calDistance1(): 5.66
calDistance2(): 5.66
```

(2) 테스트 사례 2:

```
x1 y1 x2 y2를 입력합니다:
-1 -1 5 5
calDistance1(): 8.49
calDistance2(): 8.49
```

기능을 테스트할 수 있는 샘플 프로그램은 다음과 같습니다:

```
#include <stdio.h>
#include <math.h>
void inputXY(double *x1, double *y1, double *x2, double *y2);
void outputResult(double dist);
double calDistance1(double x1, double y1, double x2, double y2);
void calDistance2(double x1, double y1, double x2, double y2, double *dist);
int main()
{
   double x1, y1, x2, y2, distance=-1;

   inputXY(&x1, &y1, &x2, &y2);                    // 참조로 호출 거리 =
   calDistance1(x1, y1, x2, y2);                   // 값으로 호출
   printf("calDistance1(): ");
   출력 결과(거리);
   calDistance2(x1, y1, x2, y2, &distance);        // 참조로 호출
   printf("calDistance2(): ");
   출력 결과(거리);                                 // 값으로 호출하면
   0을 반환합니다;
}
void inputXY(double *x1, double *y1, double *x2, double *y2)
{
   /* 여기에 코드를 작성하세요 */
}
void outputResult(double dist)
{
   /* 여기에 코드를 작성하세요 */
}
double calDistance1(double x1, double y1, double x2, double y2)
{
   /* 여기에 코드를 작성하세요 */
}
void calDistance2(double x1, double y1, double x2, double y2, double
*dist)
{
   /* 여기에 코드를 작성하세요 */
}
```