02. 포인터_실습

실습내용

- 실습#2
  - #2-1. 2차원 배열과 포인터 이해하기
  - #2-2. const 포인터 이해하기

실습#2-1 : 2차원 배열과 포인터

- 다음 프로그램을 실행시켜, 실행 결과를 통해 배열의 이름 및 주소, 배열 포인터, 이중포인터 등의 쓰임에 대해 분석해본다.
  1. 아래 코드에서 주소 값이 아닌 변수의 값이 출력되게 바꾼다.
  2. 코드는 '실습과제#1_홍길동.cpp', 실행 화면을 캡처한 사진은 '실습과제#1_홍길동.png' 으로 네이밍 한다.

```
int M[3][3]={{1,2,3},{4,5,6},{7,8,9}};

int (*ptr)[3];      int *p;      int **pt;

ptr = M;
cout << ptr << M << '□n';
cout << ptr+1 << M+1 << '□n';
cout << *(ptr+1)  << ptr[1] << *(M+1) << M[1] << '□n';   //
cout << **(ptr+1)  << **(M+1) << *M[1] << M[1][0] << '□n';

p = M[0];
cout << p << M[0] << *M << '□n';
cout << p+1 << M[0]+1 << *M+1 << '□n';
cout << *(p+1) << *(M[0]+1) << *(*M+1) << '□n';

pt = &p;   // pt = M; (X)
cout << *pt << p << '□n';
cout << **pt << *p << '□n';
```

실습#2-1 : 2차원 배열과 포인터 (정답코드)

```
#include <iostream>
using namespace std;

int main() {
    int M[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };

    int (*ptr)[3]; // 포인터 to 배열[3]
    int *p;        // 일반 정수 포인터
    int **pt;      // 이중 포인터

    ptr = M;

    // ptr, M 은 배열의 첫 주소를 가리키므로 *(ptr+0)[0]과 동일
    cout << (*ptr)[0] << " " << M[0][0] << '□n'; // 1 1
    cout << (*(ptr+1))[0] << " " << (*(M+1))[0] << '□n'; // 4 4
    cout << *(ptr[1]) << " " << *(M[1]) << " " << **(M+1) << " " << M[1][0] << '□n'; // 4 4 4 4
    cout << **(ptr+1) << " " << **(M+1) << " " << *M[1] << " " << M[1][0] << '□n';   // 4 4 4 4

    p = M[0];
    cout << *p << " " << *M[0] << " " << **M << '□n'; // 1 1 1
    cout << *(p+1) << " " << *(M[0]+1) << " " << *(*M+1) << '□n'; // 2 2 2
    cout << *(p+1) << " " << *(M[0]+1) << " " << *(*(M)+1) << '□n'; // 2 2 2

    pt = &p;
    cout << **pt << " " << *p << '□n'; // 1 1

    return 0;
}
```

실습#2-2 : const 포인터

- 아래의 3가지 경우에서의 각각의 실행이 되지 않는 코드이다. 무엇이 문제인지 각각 적으시오.

1. 텍스트 파일에 작성한 뒤 '실습과제#1_홍길동.txt'으로 네이밍 한다.
2. 실습#2-1의 파일들과 함께 압축한 뒤 과제 제출한다.

A.

```
int i1 = 10;
int i2 = 20;
const int* pInt1;

pInt1 = &i1;
*pInt1 = 30;
```

B.

```
int i1 = 10;
int i2 = 20;
int* const pInt2 = &i1;

pInt2 = &i2;
*pInt2 = 50;
```

C.

```
int i1 = 10;
int i2 = 20;
const int* const p = &i2;

p = &i1;
*p = 40;
```

실습#2-2 : const 포인터 (정답코드)

A.

```
int i1 = 10;
int i2 = 20;
const int* pInt1;

pInt1 = &i1;
// *pInt1 = 30;
// 값을 수정할 수 없으므로 제거하거나 주석 처리
```

B.

```
int i1 = 10;
int i2 = 20;
int* const pInt2 = &i1;

// pInt2 = &i2;
// ❌ 주소 변경 불가. 주석 처리 또는 제거

*pInt2 = 50;
```

C.

```
int i1 = 10;
int i2 = 20;
const int* const p = &i2;

// p = &i1;
// ❌ 주소 변경 불가
// *p = 40;
// ❌ 값 변경 불가
```

실습#2-3 : 레퍼런스

- 레퍼런스(참조) 매개 변수를 통해 평균을 리턴하고, 리턴문을 통해서는 함수의 성공 여부를 리턴하도록 아래에 제시된 average() 함수를 빈칸을 채워 완성하세요.

```
#include <iostream>
using namespace std;

bool average(int a[], int size, int& avg) {

}

int main() {
   int x[] = {0,1,2,3,4,5};
   int avg;
   if (average(_____)) cout << "평균은 " << avg << endl;
   else cout << "매개 변수 오류" << endl;

   if (average(_____)) cout << "평균은 " << avg << endl;
   else cout << "매개 변수 오류 " << endl;
}
```

```
평균은 2
매개변수 오류
```

실습#2-3 : 레퍼런스 (정답코드)

```
#include <iostream>
using namespace std;

// 평균 계산 함수 (간단 버전)
bool average(int a[], int size, int& avg) {
    if (size <= 0) return false;  // size만 확인

    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += a[i];
    }
    avg = sum / size;  // 레퍼런스를 통해 평균 전달
    return true;       // 성공 여부 리턴
}

int main() {
    int x[] = { 0, 1, 2, 3, 4, 5 };
    int avg;

    if (average(x, 6, avg))
        cout << "평균은 " << avg << endl;
    else
        cout << "매개 변수 오류" << endl;

    if (average(x, 0, avg))
        cout << "평균은 " << avg << endl;
    else
        cout << "매개 변수 오류" << endl;
}
```

```
평균은 2
매개변수 오류
```

실습#2-4 : 레퍼런스 인자 전달 함수

- find() 함수의 원형은 다음과 같다. 문자열 a에서 char c에 저장된 문자를 찾아, 해당 문자가 있는 공간에 대한 참조를 리턴 한다. 만일 변수 c에 저장된 문자를 찾을 수 없다면 success 참조 매개변수에 false를 설정하고, 찾게 되면 true를 설정한다.

```
char& find(char a[], char c, bool& success)
```

- 다음 main()이 잘 실행되도록 find()를 작성하시오.

```
int main() {
  char s[] = "Mike";
  bool b = false;
  char& loc = find(s, 'M', b);
  if(b == false){
    cout << "M을 발견할 수 없다" << endl;
    return 0;
  }
  loc = 'm'; // 'M' 위치에 'm' 기록
  cout << s << endl; // "mike"가 출력됨
}
```

실습#2-4 : 레퍼런스 인자 전달 함수(정답코드)

```
#include <iostream>
using namespace std;

char& find(char a[], char c, bool& success) {
    for (int i = 0; a[i] != '₩0'; i++) {
        if (a[i] == c) {
            success = true;
            return a[i];  // 찾았으면 그 문자의 참조 리턴 (해당
문자가 저장된 배열의 주소값을 리턴)
        }
    }

    // 못 찾았을 때: 더미 참조 반환
    success = false;
    static char dummy = '₩0';  // 정적 변수여야 참조 유효,
함수가 끝나도 안사라짐, 리턴할께 없으니 아무거나 넘겨줌
    return dummy;
}

int main() {
    char s[] = "Mike";
    bool b = false;
    char& loc = find(s, 'M', b);
    if (b == false) {
        cout << "M을 발견할 수 없다" << endl;
        return 0;
    }
    loc = 'm';  // 'M' 위치를 'm'으로 변경
    cout << s << endl;  // 출력: "mike"
}
```