실 습 문 제 제14장

1. [이중 포인터]

```c
#include <stdio.h>

int main(void) {

    int **dp, *p, i;

    i = 0;
    p = &i;
    dp = &p;

    return 0;
}
```

(a) i, p, dp의 값을 출력하는 문장을 추가하라.

(b) i, p, dp의 주소 값을 출력하는 문장을 추가하라. (a)의 결과와 어떤 관련이 있는가?

(c) p를 이용해서 변수 i의 값을 100으로 변경하는 문장을 추가하여 보자. 같은 방법으로 dp를 이용하여 변수 i의 값을 200으로 변경하여 보라.

(c) int를 double로 변경하여 (a)와 (b)를 다시 하여 보자. 어떤 차이가 있는가?

2. **[2차원 배열과 포인터] 학생들의 성적이 score 배열에 저장되어 있다. 각 학생들은 3과목을 수강하고 있다고 가정한다.**

```c
#include <stdio.h>
#define SUBJECTS 3
double column_avg(int s[][SUBJECTS], int n, int i);

int main(void) {
    int i;
    int score[][SUBJECTS] = { { 90, 76, 83 }, { 56, 90, 63  },
                              { 52, 23, 45 }, { 12, 80, 56  },
                              { 54, 35, 38 }, { 80, 80, 100 }  };
    int n = sizeof(score)/sizeof(score[0]);

    for(i = 0; i < SUBJECTS; i++)
       printf("과목 %d의 평균= %f\n", i, column_avg(score, n, i));

    return 0;
}
```

```
// 2차원 배열의 특정열의 평균을 계산
double column_avg(int s[][SUBJECTS], int n, int k)
{
    double sum = 0.0;
    int i;
    for(i = 0; i < n; i++)
        sum += s[i][k];
    return sum/n;
}
```

(a) 위의 프로그램을 컴파일하고 실행하여 화면 출력을 기록하라.

(b) 특정한 열의 평균을 구하는 column_avg() 함수를 포인터를 이용하여 다시 작성하여 보라.

(c) 특정한 학생 성적의 평균을 구하는 row_avg() 함수를 배열 첨자 방식으로 작성하여 보라.

(d) 특정한 학생 성적의 평균을 구하는 row_avg() 함수를 포인터를 이용하여 다시 작성하여 보라.

(e) 특정한 학생 성적의 평균을 구하는 함수를 1차원 배열의 평균을 구하는 함수인 get_array_avg()를 사용하여 계산하도록 적절하게 호출하라.

```
double get_array_avg(int s[], int n)
{
    double sum = 0.0;
    int i;
    for(i = 0; i < n; i++)
        sum += s[i];
    return sum/n;
}
```

3. **[포인터 배열] 포인터의 배열은 문자열을 저장하는데 가장 많이 이용된다.**

```
#include <stdio.h>
void print_strings(char *s[], int n);

int main(void) {
    int i;
    char *fruits[ ] =
    { "apple", "orange", "strawberry", "watermelon", "tomato" };

    print_strings(fruits, sizeof(fruits)/sizeof(fruits[0]));

    return 0;
```

```
}
void print_strings(char *s[], int n)
{
    // 작성하시오.
}
```

(a) 배열 첨자 방식을 이용하여 print_strings()를 작성하고 컴파일하여 실행하라.

(b) 포인터를 사용하는 방식으로 print_strings()를 작성하고 컴파일하여 실행하라.

(c) print_strings를 다음과 같이 선언하여서 똑같은 기능을 하도록 작성하여 보라.

```
void print_strings(char **s, int n);
```

(d) 다음과 같이 fruits 배열을 선언하였을 경우와 다른 점은 무엇인가?

```
char fruits[ ][20] =
{ "apple", "orange", "strawberry", "watermelon", "tomato" };
```

(e) 특정 문자열을 찾기 위하여 문자열의 배열을 탐색하는 search_strings()를 작성하여 보자. 다음과 같은 원형을 가지고 배열 첨자 방식을 이용하여 구현하라.

```
void search_strings(char *key, char *s[], int n);
```

(f) search_strings()를 포인터를 사용하여 구현하라.

4. [명령행 매개 변수] 각종 프로그램들을 실행시킬 때, 작업에 필요한 여러 가지 값들을 명령행에서 제공하는 경우가 있다.

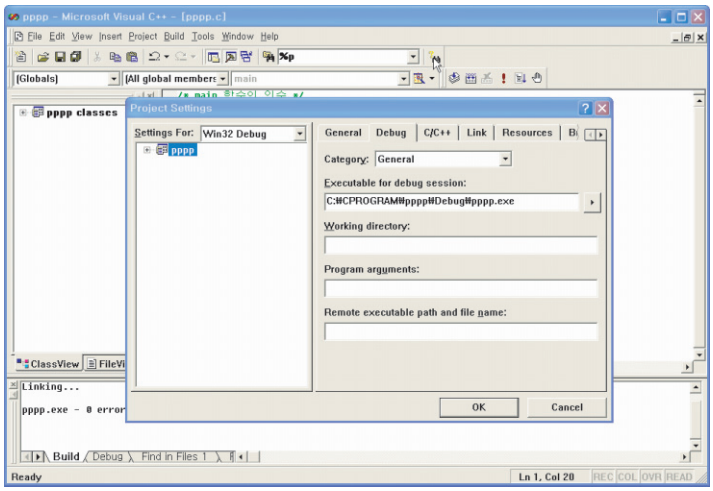
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i;

    for(i = 0; i < argc; i++)
        printf("%s\n", argv[i]);

    return 0;
}
```

(a) 위의 프로그램을 컴파일하고 실행하여서 출력 결과를 써라. 단 실행하기 전에 Project → Settings → Debug 메뉴를 선택하여서 다음 화면처럼 명령어 라인 매개 변수를 공급해주어야 한다.



(b) 위의 프로그램에서 argv[]를 분석하여서 명령행 매개 변수의 첫 번째 문자가 –로 시작하면 옵션으로 간주한다. 만약 -r이 입력되면 argv[]의 내용을 역순으로 출력하도록, 프로그램에 적절한 문장들을 추가하여 보자.

(c) 만약 -n이 입력되면 argv[]의 내용을 출력할 때 번호가 앞선 것을 먼저 출력하라.