실 습 문 제 제12장

1. **[문자와 아스키 코드] C에서는 아스키 코드라는 숫자를 이용하여 문자를 나타낸다.**

```
#include <stdio.h>

int main(void)
{
    char c;

    c = 'A';
    printf("%d %c \n", c, c);

    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(b) char c;를 int c; 로 변경하면 어떻게 되는가? 그 이유는 무엇인가?

(c) 문자 A 대신에 문자 B를 출력하도록 수정하여 보라.

(d) 문장 c = 'A';을 c = 'A'+3;로 변경하여 실행하여 보라. 어떤 문자가 출력되는가?

(e) 반복 구조를 이용하여 변수 c에 'A'부터 'z'까지를 순차적으로 대입하여 출력하도록 수정하여 보라.

2. **[문자 배열과 문자열] C에서는 문자 배열을 이용하여 문자열을 표현한다. 문자 배열에 문자들을 입력한 후에 그것을 화면에 출력하여 보자.**

```
#include <stdio.h>

int main(void)
{
    char s[10];

    s[0] = 'H';
    s[1] = 'e';
    s[2] = 'l';
    s[3] = 'l';
    s[4] = 'o';
    s[5] = '\0';
    printf("%s\n", s);

    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(b) 만약 `s[5] = '\0';`이라는 문장을 주석 처리하면 결과는 어떻게 되는가? 그 이유를 설명하라.

(c) 만약 `s[3]`에 `'l'` 대신에 0을 대입하면 어떤 결과가 생성되는가? 그 이유를 설명하라.

(d) 문자열 `"Hello"` 대신에 `"Welcome"`을 출력하도록 프로그램을 수정하여 보라.

(e) 각각의 배열 원소에 문자를 대입하지 말고, 문자 배열을 문자열 `"Hello"`로 초기화시키는 방법을 사용하여 프로그램을 다시 작성하여 보라.

3. [문자 배열의 처리] 문자 배열의 원소들을 하나씩 순차적으로 처리하는 방법을 학습하여 보자. 사용자가 입력한 문자열의 길이를 계산하여 보자.

```c
#include <stdio.h>

int main(void)
{
    int i;
    int len = 0;
    char s[80];

    printf("문자열을 입력하시오:");
    scanf("%s", s);

    for(i = 0; s[i] != 0; i++)
        len++;

    printf("문자열의 길이는 %d입니다.", len);
    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라. `"Hello"`를 입력하여 보자.

실행결과

(b) `for` 루프에서 `s[i] != 0` 대신에 `s[i]`를 사용하면 어떻게 되는가? 차이점이 있는가?

(c) 문자열의 길이를 계산하는 코드를 `my_strlen(char s[]);`의 원형을 가지는 함수로 분리하여 보라.

(d) `for` 루프를 `while` 루프로 바꾸어 보라.

4. **[문자 입출력 및 처리 함수] 문자 입출력 함수와 문자 처리 함수를 이용하여, 사용자가 입력한 문자들을 모두 대문자로 변환하여 화면에 출력하는 프로그램을 살펴보자.**

```
#include <stdio.h>
#include <ctype.h>
#include <conio.h>

int main(void)
{
    char c;
    while( (c = getchar()) != 'q' )
    {
        if( islower(c) )
            c = toupper(c);
        putchar(c);
    }

    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(b) getchar()와 putchar() 대신에 getch()와 putch()를 사용해보라. 어떤 차이점이 있는가? 헤더파일 <conio.h>를 포함하여야 한다.

(c) 사용자가 입력한 문자들을 모두 소문자로 출력하도록 프로그램을 변경하여 보라.

(d) 위의 while 루프를 무한 반복 루프인 while(1)과 break문을 사용하여 다시 작성하여 보라.

5. **[문자열 입출력 및 처리 함수] 사용자로부터 2개의 문자열을 받아서 비교하는 프로그램을 살펴보자.**

```
#include <stdio.h>

int main(void)
{
    char s1[80];
    char s2[80];

    puts("첫번째 문자열을 입력하시오:");
    gets(s1);

    puts("두번째 문자열을 입력하시오:");
    gets(s2);
```

```
    if( strcmp(s1, s2) == 0 )
        puts("두개의 문자열이 같습니다.");
    else
        puts("두개의 문자열이 같지 않습니다.");

    return 0;
}
```

(a) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(b) 처음부터 5개의 문자만을 비교하도록 strcmp() 대신에 strncmp()를 사용하도록 수정하여 보라.

(c) strcmp()를 사용하지 말고, 각 문자열을 이루는 문자들을 모두 비교하여 같은지를 검사하는 str_cmp() 함수를 직접 작성하여 실행하여 보라.

(d) strcat()을 이용하여 사용자로부터 받은 첫 번째 문자열의 뒤에 두 번째 문자열을 연결하도록 프로그램을 수정하여 보라.

6. **[문자열 수치변환] 사용자로부터 다음과 같은 문자열을 받아서, 계산을 수행하여 화면에 출력하는 프로그램을 작성하여 보자.**

(예) "10 20 add"

위의 문자열이 입력되면 "10"과 "20"을 정수로 변환한 후에 10과 20을 더하여 그 결과인 30을 화면에 출력하면 된다. 전체가 문자열로 되어 있다는 점에 주의하여야 한다. 문자열을 정수로 변환할 때 atoi() 함수를 사용하여 보자.

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

int main(void)
{
    char s1[80];
    char s2[80];
    char op[80];
    int x, y, z;

    printf("문자열을 입력하시오:\n");
    scanf("%s%s%s", s1, s2, op);

    if( strcmp("add", op) == 0 )
    {
        x = atoi(s1);
```

```
        y = atoi(s2);
        z = x + y;
        printf("연산 결과는 %d입니다\n", z);
    }

    return 0;
}
```

(m) 위의 프로그램을 컴파일하여 실행하고 그 결과를 기록하라.

실행결과

(n) "20 10 sub", "10 20 mul", "20 10 div" 등도 지원하도록 프로그램을 확장하여 보라.

(o) 무한 반복 루프를 이용하여 사용자가 "quit"라는 문자열을 입력할 때까지 반복하도록 프로그램을 확장하여 보라.

(p) atoi() 함수의 기능을 그대로 구현한 my_atoi() 함수를 작성하고 프로그램에 끼워 실행하여 보라.

(q) atoi() 대신에 sscanf()를 사용하여 다시 작성하여 보라.

(r) printf()를 사용하지 말고, 연산 결과를 새로운 문자열 result[]에 sprintf()를 이용하여 저장한 후에 puts()를 이용하여 화면에 출력하여 보라.