Chapter 1. C의 개요

연습문제

1-1. 화면에 다음 단어들을 아래 조건에 맞게 출력하여라.

She sells sea shells by the seashore

(a) 한 줄에

```
int main (void)
{
    printf("She sells sea shells by the seashore₩n");
}
```

(b) 세 줄에

```
#include <stdio.h>
int main (void)
{
    printf("She sells sea shells by the seashore₩n");
    printf("She sells sea shells by the seashore₩n");
    printf("She sells sea shells by the seashore₩n");
}
```

(c) 세 줄에

```
#include <stdio.h>
int main (void)
{
    printf("She sells sea shells by the seashore₩n");
    printf("She sells sea shells by the seashore₩n");
    printf("She sells sea shells by the seashore₩n");
}
```

1-2. 1.3절에 있는 marathon 프로그램의 결과가 옳은지 계산기를 사용하여 확인하여라. 이 프로그램에서 실수형 상수 1760.0을 정수형 상수 1760으로 수정하여라. 이 수정된 프로그램을 컴파일한 후 실행하여 그 결과가 수정하기 전의 것과 같지 않음을 확인하여라. 정수 나눗셈은 소수점이하를 버리기 때문에 그 결과가 달라진다.

```
[romance@161s ch1]$ cc 1_2.c
[romance@161s ch1]$ ./a.out

A marathon is 42.185970 kilometers.

[romance@161s ch1]$ vi 1_2.c
#include <stdio.h>
int main(void)
{
  int miles,yards;
  float kilometers;
  miles = 26;
  yards = 385;
  kilometers = 1.609*(miles+ yards / 1760);          // 1760.0을 1760으로 변경.
  printf("₩nA marathon is %f kilometers. ₩n₩n",kilometers);
  return 0;
}
~
[romance@161s ch1]$ cc 1_2.c
[romance@161s ch1]$ ./a.out

A marathon is 41.834000 kilometers.

[romance@161s ch1]$
```

1-3. marathon 프로그램의 모든 상수와 변수를 double 형으로 수정하여라. 이 수정된 프로그램의 결과가 원래 프로그램의 결과와 같은가?

```
#include <stdio.h>
int main(void)
{
  double miles,yards;                    // double 형으로 변경
  double kilometers;
  miles = 26;
  yards = 385;
  kilometers = 1.609*(miles+ yards / 1760);
  printf("₩nA marathon is %f kilometers. ₩n₩n",kilometers);
  return 0;
}
[romance@161s ch1]$ cc 1_3.c
[romance@161s ch1]$ ./a.out

A marathon is 42.185969 kilometers.

[romance@161s ch1]$
```

수정된 결과는 원래 프로그램과 일치함을 알 수 있다.

1-4. marathon 프로그램의 다음문장에서 void를 삭제하여라.

int main(void)

이 수정된 프로그램을 컴파일 하여라. 컴파일 할 때 경고 메시지가 발생하는가? 아마 그렇지 않을 것이다.

그다음은 main() 함수의 몸체에서 다음 문장을 삭제하여라.

Return 0;

이 수정된 프로그램을 컴파일 해보고 컴파일 중에 경고 메시지가 발생하는지 확인하여라. 일반적으로, 프로그래머는 컴파일러를 가장 높은 경고 단계로 설정한 후 사용 하는 것이 좋다. 프로그래밍의 원칙 중 하나는 “컴파일러가 경고 메시지를 발생하지 않게 하여라” 이다. 따라서 프로그래머는 모든 경고들이 제거 될 때 까지 계속해서 코드를 수정 보완해야 한다.

1) void를 없앤 후 결과

```
[romance@161s ch1]$ cc 1_4.c
[romance@161s ch1]$ ./a.out

A marathon is 42.185970 kilometers.

[romance@161s ch1]$
```

2) return 0; 을 없앤 후 결과

```
[romance@161s ch1]$ cc -Wall 1_4.c
1_4.c: In function `main':
1_4.c:11: warning: control reaches end of non-void function
[romance@161s ch1]$
```

Additonal Text

컴파일러의 옵션

컴파일러 동작 표시하기.

-v : 각 단계에서 실행하는 자세한 사항을 출력. 어떤 옵션으로 컴파일하였는지 알아낼 때 도움이 된다.

C 언어 옵션:

-w : 모든 경고메시지가 나오지 않도록 한다.

-W : 합법적이지만 모호한 코딩에 대하여 부가적 경고메지시출력

-Wall : 모호한 코딩에 대하여 훨씬 자세한 경고메시지를 출력.
(항상 이 옵션을 사용하도록 하자. 기계가 할 수 있는 일은 기계에게 맡기는 것이다.)

전처리기 옵션(cpp 전처리기 제어)

-C : -E 옵션과 함께 써야 하며, 프로그램의 주석을 지우지 않는다.

라이브러리 지정옵션:

-static : 공유라이브러리가 아닌 정적 라이브러리와 링크한다.

-shared : 공유라이브러리와 링크한다 가능한.(기본값)

1-5. 다음 프로그램은 실행시간 오류를 유발할 것이다.

```
#include <stdio.h>
int main(void)
{
  int x,y = 0;
  x = 1/y ;
  printf("x=%d₩n",x);
  return 0;
}
```

이 프로그램을 컴파일해 보아라. 아마 아무런 오류 메시지도 발생하지 않을 것이다. 이 프로 그램을 실행시켜보면, 영으로 정수를 나누는 것이 어떤 결과를 가져 오는 지를 알 수 있을 것이다. 대부분의 시스템에서 이 프로그램은 실행시간 오류를 발생시킨다.

```
[romance@161s ch1]$ cc 1_5.c
[romance@161s ch1]$ ./a.out                    // 실행
Floating point exception
[romance@161s ch1]$
```

만일 여러분의 시스템 상에서 실행시간 오류가 발생했다면, 이 프로그램에서 변수 y를 사용하지 말고 직접 영을 사용하여 나누기를 해 보아라. 이제 어떤 일이 일어나는가?

```
[romance@161s ch1]$ cc -Wall 1_5_a.c          // -Wall 옵션.
[romance@161s ch1]$ ./a.out                   // 실행
Floating point exception
[romance@161s ch1]$
```

1-6. 대부분의 C시스템들은 논리적으로 무한한 실수값들을 제공한다. 연습문제 5번 프로그램에서 int를 double로 수정하고, printf() 문에서 %d 를 %f 로 수정하여라. 그 다음 이 프로그램을 수행하면 실행시간 오류가 발생할까? 대부분의 시스템들 에서는 실행시간 오류가 발생하지 않을 것이다. 자신의 시스템에서는 어떻게 되는 지 수행시켜 보아라.

```
[romance@161s ch1]$ cc 1_6.c
[romance@161s ch1]$ ./a.out
x=0                                    // floating point exception error가 없음.
[romance@161s ch1]$
```

1-7. 일반적으로 프로그램에서는 모든 #include 행들은 파일의 시작부분에 온다. 그러나 반드시 시작 부분에 위치해야만 하는가? 1.4절의 pacific_sea 프로그램에서 #include 행을 다음과 같이 main()의 몸체에 오도록 수정해 보아라.

```
 int main(void)
{
   #include “pacific_sea.h”
   …….
```

이 수정된 프로그램을 실행시켜 보아라.

```
[romance@161s ch1]$ cc 1_7.c
[romance@161s ch1]$ ./a.out

The Pacific Sea covers an areaof 2337 square kilometers.
 In other units of measure this is:

      5.7748528e+ 05 acres
      9.0232074e+ 02 square miles
      2.5155259e+ 10 square feet
      3.6223572e+ 12 square inches
```

1-8. sea.c 프로그램을 sea로 파일 변경한 후에 컴파일 해보아라 이렇게 컴파일하면, 어떤 컴파일러는 경고 메시지를 낸다. UNIX 시스템에서는 “bad magic number” 또는 “unable to process using elf libraries”와 같은 메시지가 발생하기도 한다. 여러분의 시스템은 어떤 메시지를 출력하는가?

```
[romance@161s ch1]$ cc –Wall sea
/usr/bin/ld:1_7: file format not recognized; treating as linker script
/usr/bin/ld:1_7:2: parse error
collect2: ld returned 1 exit status
[romance@161s ch1]$
```

1-9. 다음은 화면에 대문자 I를 출력하는 프로그램이다.

```
#include <stdio.h>
#define BOTTOM_SPACE "₩n₩n₩n₩n₩n"
#define HEIGHT          17
#define OFFSET          "                    "
#define TOP_SPACE "₩n₩n₩n₩n₩n"
int main(void)
{   int i;
    printf(TOP_SPACE);                          // 상부 5행을 띤다.
    printf(OFFSET "IIIIIII₩n");                 // I의 윗부분을 그린다.
    for(i=0;i<HEIGHT;++i)                       // I의 높이인 17만큰 중간부분을 그린다.
     printf(OFFSET "   III₩n");
    printf(OFFSET "IIIIIII₩n");                 // I의 아랫부분을 그린다.
    printf(BOTTOM_SPACE);                       // 하부 5행을 띤다.
    return 0;
}
```

```
[romance@161s ch1]$ ./a.out



              IIIIIII
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
                III
              IIIIIII
```

이 프로그램을 실행시켜 그 결과를 써라. 그리고 이 프로그램을 분석한 후에 화면에 대문자 C를 출력하는 프로그램을 작성하여라.

```
// 1_9_a.c   대문자 C 를 출력하는 프로그램.
#include <stdio.h>
#define BOTTOM_SPACE "₩n₩n₩n₩n₩n"
#define HEIGHT          4
#define OFFSET          "                    "
#define TOP_SPACE "₩n₩n₩n₩n₩n"
int main(void)
{   int i;
    printf(TOP_SPACE);
    printf(OFFSET "   CCCCC₩n");
    printf(OFFSET " C       ₩n");
    for(i=0;i<HEIGHT;++i)
     printf(OFFSET "C₩n");
    printf(OFFSET " C₩n");
    printf(OFFSET "   CCCCC₩n");
    printf(BOTTOM_SPACE);
    return 0;
}
```

1-10. 실행이 되는 프로그램을 선택하여 한 행씩 지워가며 컴파일해 보아라. 한 행을 지운 다음 컴파일 할 때 발생하는 오류 메시지를 모두 기억하여라. 예를 들어 다음 프로그램을 사용해 보아라.

```
#include <stdio.h>
{
printf("nonsense₩n");
}

[romance@161s ch1]$ cc -Wall 1_10.c
1_10.c:2: parse error before `{'
[romance@161s ch1]$
```

1-11. name과 age를 받아들인 후 다음 문장을 출력하는 프로그램을 작성하여라.

Hello *name*, next year you will be *next_age*.

```
#include <stdio.h>

int main(void)
{
   char name[80];
   int age;
   printf("Input your Name ");                    // 이름을 입력받는다.
   scanf("%s",&name);
   printf("Input your Age ");                     //  나이를 입력받는다.
   scanf("%d",&age);
   printf("₩n Hello %s, next year you will be %d ₩n₩n",name,age+ 1); //결과 출력
   return 0;
}
```

1-12. 각 정수에 대한 거듭제곱을 다음 표와 같이 출력하는 프로그램을 작성하여라.

::::: A TABLE OF POWERS :::::

Integer	Square	3rd power	4th power	5th power
1	1	1	1	1
2	4	8	16	32
3	9	27	81	243

```
#include <stdio.h>
int main(void)    {   int i;
printf("::::: A TABLE OF POWERS :::::₩n");
printf("Integer Square   3rd power         4th power         5th power₩n");
printf("_____   _____   _______          _______          _______₩n");
for(i=1;i<4;i+ + )
printf("%5d          %5d          %5d                        %5d                        %5d
₩n",i,i*i,i*i*i,i*i*i*i,i*i*i*i*i);
printf("₩n============================================================₩n");
return 0;}
```

1-13. for 루프의 형식은 다음과 같다.

For(expr1;expr2;expr3)

Statement

만일 세 수식을 모두 사용한다면, 이 문장은 다음과 같다.

Expr1;

While(expr2) {

Statement

expr3;

}

왜 statement 다음에는 세미콜론이 없을까? 물론 statement 자체는 맨 끝에 세미콜론을 포함하는 경우가 대부분이다. 하지만 반드시 그렇지는 않다. C에서 복합문은 중괄호에 둘러싸여 있는 0 개 이상의 다른 문장들로 구성되고, 복합문 그 자체도 하나의 문장이다. 따라서 { }와 { ; ; ; }도 문장이다. 다음 코드를 포함하는 프로그램을 작성해 보아라.

```
Int i;
For (i=0; i<3; + + i)
{}
For(i=0;i<3;+ + i)
{ ; ; ; }
```

```
#include <stdio.h>

int main(void)
{
  int i;
  for (i=0; i<3; + + i)
  {}
  for( i=0; i<3; + + i)
  { ; ; ; }
}
```

1-14. C 시스템이 제공하는 표준헤더 파일들은 하나 이상의 디렉토리에 있을 수 있다. 예를 들어, UNIX 시스템에서 헤더 파일들은 /usr/include에 있다. Turbo C에 서는 ₩turbo₩include ₩, ₩tc₩include, 또는 ₩bc₩include 디렉토리에 있다. 여러분이 사용하는 시스템에서 표준 헤더 파일 stdio.h가 어디에 있는지 찾아보아라. 그리고 이 파일에서는 printf() 함수가 선언된 행을 찾아 보아라. 그 행은 다음과 같은 것이다.

```
int printf(const char *format, ....);
```

이 행은 함수 원형의 예이다. 함수 원형은 함수에 전달될 인자의 개수와 형, 그리고 함수에 의해 리턴될 값의 형을 컴파일러에게 알려준다. 앞으로 설명하겠지만, 문자열은 char 포인터이며 char *로 나타낸다. Format은 단지 프로그래머가 쉽게 이해할 수 있도록 하는 것으로, 컴파일러는 이것을 무시한다. 따라서 printf() 함수 원형은 다음과 같이 쓸수도 있다.

```
int   printf(const char*, .... );
```

const 키워드는 printf() 함수의 인자로 전달되는 문자열이 변경될 수 없음을 컴파일러에게 알려준다. 생략부호 ...는 나머지 인자들의 개수와 형이 변할 수 있음을 알려 준다. printf() 함수는 전달된 문자의 개수를 int 형으로 리턴하고, 오류가 발생하면 음수값을 리턴한다. 이 장의 첫번째 예제인 sea.c에서 #include 문을 지우고 그 대신 위에 있는 printf() 함수 원형을 써라. 이렇게 수정한 프로그램을 컴파일하고 실행시켜 보아라.

```
int printf(const char*, ...);

int main(void)
{
   printf("from sea to shining C₩n");
   return 0;
}
[romance@161s ch1]$ cc 1_14.c
[romance@161s ch1]$ ./a.out
from sea to shining C
[romance@161s ch1]$
```

1-15. (스텐포드 대학의 Donald Knuth가 제안한 문제) 1.6절의 running_sum 프로그램에서 평균값을 계산하기 위해 먼저 숫자의 합을 구하고 더해진 숫자의 개수로 그 합을 나누었다. 다음 프로그램 better_average는 평균값을 더 좋은 방법을 사용하여 계산한 것이다.

```
#include <stdio.h>
int main(void)
{
    int i;
    double x;
    double avg = 0.0;
    double navg;
    double sum = 0.0;

printf("%5s%17s%17s%17s₩n%5s%17s%17s%17s₩n₩n","Count","Item","Average","Naive
avg","-----","----","-------","----------");
    for(i=1;scanf("%lf",&x) == 1; + + i) {
    avg + = (x - avg)/i;
    sum+ =x;
    navg = sum /i;
    printf("%5d%17e%17e%17e₩n",i,x,avg,navg);
    }
    return 0;
}
```

이 프로그램을 수행시켜 보고 원리를 이해하여라. 이 새로운 알고리즘은 평균값을 계산하기 위해 다음 행을 사용하고 있다.

avg + = (x-avg) / i;

이 알고리즘이 어떻게 평균값을 구하게 되는지 설명하여라.

Sol) avg 라는 것은 전체 값들을 합한 후에 값들의 개수로 나눈 값이다.

즉 avg를 계산하기 전의 sum 값은 (전의 avg) * (i-1) 이고 이를 x 값을 더하면 sum 이 된다. 즉 새로운 avg값은 {(전의 avg)*(i-1)+ x} / i 인 셈이다. 따라서 따라서 이 식들을 정리하면 {(전의 avg)* i + x –avg }/ i 인 것이다. 이를 다시 나타내면 avg + =(x-avg) / i 가되는 것이다.

1-16. 이 연습문제는 연습문제 15번의 better_average 프로그램에서 sum이 컴퓨터에서 표현 가능한 값보다 더 큰 값을 가질 때 어떤 일이 발생하는 지를 알아본다.
Data라는 파일을 만들어 다음 숫자들을 그 파일에 써넣어라.
1e308 1 1e308 1 1e308
그 다음 data 파일을 better_average 의 입력파일로 재지정한 후 수행시켜 보아라. 이 알고리즘의 장점을 알겠는가?

```
[romance@161s ch1_code]$ ./a.out < 1_16.dat
Count              Item           Average         Naive avg
-----              ----           -------         ---------

    1     1.000000e+308     1.000000e+308     1.000000e+308
    2      1.000000e+00     5.000000e+307     5.000000e+307
    3     1.000000e+308     6.666667e+307              inf
    4      1.000000e+00     5.000000e+307              inf
    5     1.000000e+308     6.000000e+307              inf
[romance@161s ch1_code]$
```

1-17. 이 연습문제는 연습 문제 16번에서 계속되는 것이다. better_average 프로그램의 입력으로 위와 같은 큰 숫자가 아닌 평범한 숫자를 사용한다면, Average 열과 Naïve avg 열은 같을 것 처럼 보인다. 두 열의 값이 달라지는 경우를 찾아 보아라. 즉, sum이 오버플로 되지 않을 때에도 better_average 가 실제로 더 좋은 방법이라는 것을 실험적으로 증명하여라.

(clear)

1-18. 이 연습 문제는 형 한정자인 const에 관한 것이다. 컴파일러는 다음 코드를 어떻게 다루겠는가?

```
const int a = 0;
a = 333;
printf (“%d₩n”,a);
```

```
[romance@161s ch1_code]$ cc 1_17.c
1_17.c: In function `main':
1_17.c:6: warning: assignment of read-only variable `a'
[romance@161s ch1_code]$ ./a.out
333
[romance@161s ch1_code]$
```

1-19. 다음 코드를 포함하는 프로그램을 작성한 후 수행시켜 보아라.

```
int a1,a2,a3,cnt;
printf (“Input three integers : “);
cnt = scan(“%d%d%d”,&a1,&a2,&a3);
printf (“Number of successful conversion : %d₩n”,cnt);
```

프로그램이 입력을 기다릴 때 문자 x를 입력하면 어떻게 되는가?

```
Input three integers : x
Number of successful conversion : 0
[romance@161s ch1_code]$
```

1-20. ANSI C에서 printf() 함수는 출력한 문자의 개수를 int형으로 리턴한다 이것이 어떻게 작동하는지 알아보기 위해 다음 코드를 포함하는 프로그램을 작성하여라.

```
int cnt;
cnt = printf("abc abc");
printf ("₩n No. of characters printed :  %d₩n",cnt);
```

이 프로그램의 결과는 무엇인가? 만일 첫번째 printf()의 제어 문자열을 “abc₩nabc ₩n” 또는 “abc₩0abc₩0”으로 수정한다면 무엇이 출력되겠는가?

```
1) “abc abc”
abc abc
 No. of characters printed :  7
[romance@161s ch1_code]$

2) “abc₩nabc ₩n”
[romance@161s ch1_code]$ ./a.out
abc
abc

 No. of characters printed :  9
[romance@161s ch1_code]$

3) “abc₩0abc₩0”
abc
 No. of characters printed :  3
[romance@161s ch1_code]$
```

1-21 연습문제 19번의 프로그램은 수행결과가 입력에 따라 다르게 나왔다. 다음 코드를 포함하는 프로그램을 작성하여라.

```
char c1,c2,c3;
int cnt;
printf(“Input three characters : “);
cnt = scanf(“%c%c%c”,&c1,&c2,&c3);
printf (“Number of successful conversions : %d₩n”,cnt);
```

```
[romance@161s ch1_code]$ ./a.out
Input three characters : abc
Number of successful conversions : 3
[romance@161s ch1_code]$ ./a.out
Input three characters : 123
Number of successful conversions : 3
[romance@161s ch1_code]$ ./a.out
Input three characters : aprkjasdfklj
Number of successful conversions : 3
[romance@161s ch1_code]$ ./a.out
Input three characters : %^&
Number of successful conversions : 3
[romance@161s ch1_code]$ ./a.out
Input three characters :
Number of successful conversions : 3
[romance@161s ch1_code]$ ./a.out
Input three characters : d
d
Number of successful conversions : 3
[romance@161s ch1_code]$
```

1-22. 1.8절의 nice_day 프로그램에서 사용한 아이디어를 이용하여 총 문자 개수를 계산하는 nletters 프로그램을 작성하여라. 프로그램을 수행할 때 재지정을 사용하여라. Infile이 문서 파일이라면, 다음과 같이 프로그램을 수행한다.

nletters < infile
또한 다음과 같이 출력되도록 하여라.
Number of letters : 179

C code

```
#include <stdio.h>
#include <ctype.h>
#define MAXSTRING   100
int main(void)
{
   char c,name[MAXSTRING];
   int i,sum=0;
   int num=0;
   for (i=0; (c=getchar()) != '₩n' ; + + i)
   {
    if (isalpha(c)) + + num;
   }
   printf("Number of letters : %d₩n",num);

   return 0;
}
```

Result (결과)

```
[romance@161s ch1_code]$ ./a.out
park
Number of letters : 4
[romance@161s ch1_code]$ ./a.out   < 1_22.dat
Number of letters : 7
[romance@161s ch1_code]$
```

1-23. 1.8절의 abc프로그램에서 다음 루프를 사용하였다.

```
for ( ; *p != '₩0' ; + + p) {
  if  (**p =='e')
      *p = 'E';
  if (*p == ' ')
       *p = '₩n';
  }
```

이 for 루프는 두 개의 if문으로 구성되기 때문에 중괄호를 사용하였다. 이 코드를 다음과 같이 수정하였다.

```
for ( ; *p != '₩0' ; + + p)
  if  (**p =='e')
      *p = 'E';
  else if (*p == ' ')
       *p = '₩n';
```

수정된 코드는 왜 중괄호가 필요 없는지 설명해 보아라. 이렇게 수정된 프로그램도 이전 프로그램의 결과와 같은지 살펴보아라. 같다면 그 이유를 설명하여라.

Sol) for 다음 statement는 괄호가 없을 경우에는 1 문장만을 반복하는데 중괄호 없이 밑의 코드로 고쳤을 경우는 if ~ else if 를 모두 같은 문장으로 보기 때문에 위의 두 코드는 같은 결과를 출력한다.

1-24. a가 임의의 형의 배열이고 i는 int형이라고 하자 . 이때 a[i]를 이와 같은 의미를 갖는 포인터 수식으로 표현할 수 있다. 그 포인터수식을 써라.

Sol) a[i] == *a + i

예를 들어 *a는 a[0]을 가르키는 포인터인 셈이다.

1-25. 다음 프로그램에는 prn_string() 함수가 정의 되어 있지않다. 인자로 전달되는 문자열을 출력하기 위해 putchar()를 사용하는 prn_string() 함수를 작성하여 프로그램을 완성하여라. 문자열은 널 문자 ₩0 으로 끝난다는 것을 기억하여라. 이 프로그램은 표준 라이브러리에 있는 strcat()함수를 사용한다. 이 함수의 원형은 string.h 헤더 파일에 있다. 이 함수는 두개의 문자열을 인자로 가지며, 이 두 문자열을 연결한 다음 그 결과를 첫번째 인자에 넣는다.

C code

```
#include <stdio.h>
#include <string.h>
#define MAXSTRING 100
void prn_string (char *);
int main(void)
{ char s1[MAXSTRING],s2[MAXSTRING];
   strcpy (s1,"Mary, Mary, quite contrary, ₩n");
   strcpy (s2,"how does your garden grow?₩n");
   prn_string(s1);
   prn_string(s2);
   strcat(s1,s2);
   prn_string(s1);
   return 0;          }
void prn_string (char *a)
{   int i=0;
   while (a[i] != '₩0')   {
   putchar (a[i]);   ++i;   }}
```

Result (결과)

```
[romance@161s ch1_code]$ ./a.out
Mary, Mary, quite contrary,
how does your garden grow?
Mary, Mary, quite contrary,
how does your garden grow?
[romance@161s ch1_code]$
```

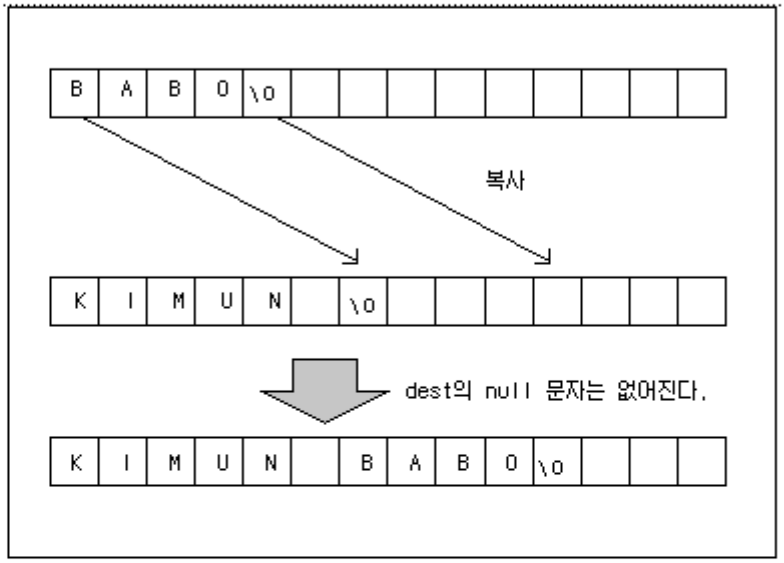
strcat

DOS	UNIX	WINDOWS	
○	○	○	2급

원형 char *strcat(char *dest, const char *src);

헤더 파일 string.h

기능 문자열끼리 연결한다. dest 뒤에 src가 덧붙여지며 dest 뒤의 NULL 문자는 없어지고 src의 NULL 문자는 남는다. dest가 "KIMUN"이고 src가 "BABO"일 때 strcat(dest, src)에 의해 dest는 "KIMUN BABO"가 된다.



dest는 자신이 가지고 있는 문자열 길이 외에도 src의 문자열 길이만큼을 더할 수 있는 충분한 크기이어야 한다. 만약 그렇지 못할 경우 인접한 데이터가 파괴된다.

리턴값 dest의 번지가 리턴된다.

참고 함수 strncat ⇨ 특정 개수의 문자열만 추가시킨다.

본문 참조 C++을 내것으로 문법편 12-6절 참조

예제 다음 예제는 위에서 보인 두 문자열을 합치는 예제이다.

```
#include <stdio.h>
#include <string.h>
void main()
{
char src[20];
char dest[20];
strcpy(src,"BABO");
strcpy(dest,"KIMUN ");
strcat(dest,src);
printf("destination string is %s\n",dest);
}
```

1-26. 파일 재지정을 실험해 보자. 1.10절 “운영체제의 고찰”에서 dbl_out이라는 프로그램이 있었다. 몇줄의 문장으로 구성된 my_file이라는 파일을 만들어라. 재지정의 효과를 이해하기 위해 다음 명령을 사용하여 프로그램을 실행시켜 보아라.

Dbl < my_file
Dbl_out <my_file >tmp

다음 명령의 동작은 재미있다.
Dbl_out >tmp

이 명령은 dbl_out이 키보드로부터 입력을 받아 그 결과를 tmp 파일에 쓰게 한다. 물론 입력을 끝낼 때 파일의 끝을 알려주는 기호를 입력해야 한다. 캐리지 리턴 키 다음에 control-d를 입력하지 않고, 프로그램을 끝내는 control-c를 입력하면 어떤일이 발생하겠는가? tmp 파일에는 어떤 것이 출력되어 있겠는가?

1_22.dat 파일 내용 ::
bcdadsf
park hojin is chunjea~!!

Result (결과)

```
[romance@161s ch1_code]$ ./a.out < 1_22.dat
bbccddaaddssff

ppaarrkk   hhoojjiinn   iiss   cchhuunnjjeeaa~~!!!!


[romance@161s ch1_code]$ ./a.out < 1_22.dat   > tmp
[romance@161s ch1_code]$

[romance@161s ch1_code]$ vi tmp
bbccddaaddssff

ppaarrkk   hhoojjiinn   iiss   cchhuunnjjeeaa~~!!!!
```

[romance@161s ch1_code]$ ./a.out > tmp
parkhojin is park
..
(control – d)
[romance@161s ch1_code]$ vi tmp
ppaarrkkhhoojjiinn iiss ppaarrkk

....

[romance@161s ch1_code]$ ./a.out > tmp
parkhojin is park
..
(control-c)
[romance@161s ch1_code]$ vi tmp

(아무 내용도 없음)

Chapter 1 End point.