**CHAPTER 06 함수 예제**

// Ex06\_02

#include <stdio.h>

void PrintSumAndAverage(int a, int b) {

printf("합계: %d\n", a + b);

printf("평균: %f\n", (double)(a + b) / 2);

}

int main(void) {

int x, y;

PrintSumAndAverage(10, 20);

printf("두 정수를 입력하세요: ");

scanf("%d %d", &x, &y);

PrintSumAndAverage(x, y);

return 0;

}

**<Ex06\_02 실행결과>**

**합계: 30**

**평균: 15.000000**

**두 정수를 입력하세요:** 15 20

**합계: 35**

**평균: 17.500000**

**Program ended with exit code: 0**

// Ex06\_03

#include <stdio.h>

int GetFactorial(int num) {

int i;

int fact = 1;

for(i = 1; i<= num; i++)

fact \*= i;

return fact;

}

int GetSum(int num) {

int i;

int sum = 0;

for(i = 1; i <= num; i++)

sum += i;

return sum;

}

int main(void) {

int result1, result2;

result1 = GetFactorial(10);

printf("10 팩토리얼 = %d\n", result1);

result2 = GetSum(10);

printf("1~10의 합계 = %d\n", result2);

return 0;

}

**<EX 06\_03 실행결과>**

**10 팩토리얼 = 3628800**

**1~10의 합계 = 55**

**Program ended with exit code: 0**

// Ex06\_04

#include <stdio.h>

double GetMax(double a, double b, double c) {

double max;

max = a > b ? a : b;

max = c > max ? c : max;

return max;

}

int main(void) {

double x, y, z;

double result;

printf("세 개의 실수를 입력하세요: ");

scanf("%lf %lf %lf", &x, &y, &z);

result = GetMax(x, y, z);

printf("result = %f\n", result);

// result = GetMax(0.5, 10.5);

// result = GetMax(0.5, 10.5, 12.5, 1.5);

return 0;

}

**<EX 06\_04 실행결과>**

**세 개의 실수를 입력하세요:** 1.2 3.4 0.6

**result = 3.400000**

**Program ended with exit code: 0**

// Ex 06\_06

#include <stdio.h>

int GetFactorial(int num);

double GetMax(double a, double b, double c);

int main(void) {

int i\_res;

double f\_res;

i\_res = GetFactorial(5);

printf("5! = %d\n", i\_res);

f\_res = GetMax(0.5, 10.5, 12.5);

printf("최대값 = %f\n", f\_res);

return 0;

}

int GetFactorial(int num) {

int i;

int fact = 1;

for(i = 1; i<= num; i++)

fact \*= i;

return fact;

}

double GetMax(double a, double b, double c) {

double max;

max = a > b ? a : b;

max = c > max ? c : max;

return max;

}

**<EX 06\_06 실행결과>**

**5! = 120**

**최대값 = 12.500000**

**Program ended with exit code: 0**

// Ex 06\_08

#include <stdio.h>

void PrintCount(void);

int main(void) {

int count = 0;

printf("main: count = %d\n", count);

PrintCount();

return 0;

}

void PrintCount(void) {

int count = 100;

printf("PrintCount: count = %d\n", count);

}

**<EX 06\_08 실행결과>**

**main: count = 0**

**PrintCount: count = 100**

**Program ended with exit code: 0**

// Ex 06\_10

#include <stdio.h>

void PrintCount(void);

void Increment(void);

void Decrement(void);

int count;

int main(void) {

count = 0;

PrintCount();

Increment();

Increment();

PrintCount();

Decrement();

PrintCount();

return 0;

}

void PrintCount(void) {

printf("count = %d\n", count);

}

void Increment(void) {

count++;

}

void Decrement(void) {

count--;

}

**<EX 06\_10 실행결과>**

**count = 0**

**count = 2**

**count = 1**

**Program ended with exit code: 0**

// Ex06\_13

#include <stdio.h>

void Test(void);

int num = 10;

int main(void) {

int num = 20;

while(1) {

int num = 30;

printf("num = %d\n", num++);

if(num > 25)

break;

}

printf("num = %d\n", num);

Test();

return 0;

}

void Test(void) {

printf("num = %d\n", num);

}

**<EX 06\_13 실행결과>**

**num = 30**

**num = 20**

**num = 10**

**Program ended with exit code: 0**

// Ex06\_14

#include <stdio.h>

void Swap(int x, int y);

int main(void) {

int a = 10;

int b = 20;

printf("Swap 전의 a = %d, b = %d\n", a, b);

Swap(a, b);

printf("Swap 후의 a = %d, b = %d\n", a, b);

return 0;

}

void Swap(int x, int y) {

int temp;

temp = x;

x = y;

y = temp;

}

**<EX 06\_14 실행결과>**

**Swap 전의 a = 10, b = 20**

**Swap 후의 a = 10, b = 20**

**Program ended with exit code: 0**

//Ex 06\_15

#include <stdio.h>

void GetSmallerAndLarger(int a, int b, int smaller, int larger);

int main(void) {

int smaller = 0, larger = 0;

GetSmallerAndLarger(10, 20, smaller, larger);

printf("smaller = %d, larget = %d\n", smaller, larger);

return 0;

}

void GetSmallerAndLarger(int a, int b, int smaller, int larger) {

smaller = a < b ? a : b;

larger = a > b ? a : b;

}

**<EX 06\_15 실행결과>**

**smaller = 0, larget = 0**

**Program ended with exit code: 0**

**CHAPTER 06 연습문제**

1. 함수에 대한 설명 중 잘못된 것을 모두 고르시오.

① 함수를 호출하려면 반드시 함수의 선언 또는 정의가 필요하다.

② 함수의 정의는 생략할 수 없다.

③ 함수 선언 시 매개변수 이름은 생략할 수 없다.

④ 함수를 호출할 때는 매개변수의 개수와 인자의 개수가 같아야 한다.

⑤ 리턴 값을 갖지 않는 함수는 리턴 값을 생략한다.

2. 함수를 사용할 때의 장점이 아닌 것은?

① 코드가 중복되지 않고 간결해진다.

② 코드의 재사용성이 증가된다.

③ 프로그램의 모듈화가 증대된다.

④ 프로그램의 기능을 함수 단위로 나누어 두었기 때문에 코드를 수정하기가 어려워진다.

3. 함수의 리턴형과 매개변수에 대한 설명 중 잘못된 것을 모두 고르시오.

① 함수의 리턴형은 함수가 처리 결과로 리턴하는 값의 데이터형이다.

② 함수의 리턴형을 여러 개 지정할 수 있다.

③ 함수의 매개변수는 함수 안에 선언된 변수를 말한다.

④ 함수의 매개변수의 개수에는 제한이 없다.

⑤ 함수 호출 시 넘겨준 인자는 매개변수로 전달된다.

⑥ 함수의 매개변수도 일종의 지역변수이다.

4. 다음은 함수의 호출 과정이다. 올바른 순서로 나열하시오.

A. 함수의 매개변수로 인자를 전달한다.

B. 함수 호출문 다음의 코드를 수행한다.

C. 인자가 수식인 경우 수식의 값을 평가한다.

D. 함수의 코드를 수행한다.

E. 함수의 정의로 이동한다.

F. 함수를 호출한 곳으로 되돌아온다.

G. 함수를 호출한 곳으로 리턴 값을 전달한다.

C → A → E → D → G → F → B

5. 지역 변수에 대한 설명 중 잘못된 것을 모두 고르시오.

① 함수 밖에 선언된 변수를 지역 변수라고 한다.

② 지역 변수는 따로 초기화하지 않으면 쓰레기 값이다.

③ 서로 다른 함수에서는 같은 이름의 지역 변수를 선언할 수 있다.

④ 다른 함수에 선언된 지역 변수를 사용할 수 있다.

⑤ 함수 안에서 다시 블록을 정의하고 블록 안에 지역 변수를 선언할 수 있다.

6. 같은 이름의 변수가 여러 번 선언되었을 때 가장 가까운 블록 안에 선언된 변수가

우선적으로 사용되는 규칙을 무엇이라고 하는가?

영역 규칙(scope rule)

7. 함수 바깥쪽에 선언되어 프로그램 전체에서 사용될 수 있는 변수를 무엇이라고 하

는가?

전역 변수

8. 함수의 인자를 전달할 때 인자의 값을 매개변수로 복사해서 전달하는 방식은?

값에 의한 전달(Passing by value)

9. 다음 중 올바른 함수 선언이 아닌 것은?

① int Get Sum(int num);

② void PrintHello();

③ GetFactorial(int num);

④ double GetMax(double, double, double);

10. 함수의 인자 전달 방법에 대한 설명 중 잘못된 것은?

① 함수를 호출할 때 넘겨주는 값을 실매개변수 또는 인자라고 한다.

② 함수 호출 시 전달되는 값을 받아오기 위한 변수를 형식 매개변수 또는 매개변수라고

한다.

③ 함수의 형식 매개변수는 함수 호출 시 생성되는 지역변수이다.

④ 함수를 호출할 때 인자가 매개변수로 복사되어 전달된다.

⑤ 함수가 리턴해도 함수의 매개변수를 해제되지 않고 남아있는다.

11. 다음 프로그램에서 \_\_\_\_\_\_\_\_\_\_\_\_\_에 필요한 코드를 작성하시오.

#include <stdio.h>

#include <math.h>

int main()

{

int a, b, c, d;

double length;

printf("직선의 시작점의 좌표 x, y를 입력하세요 : ");

scanf("%d %d", &a, &b);

printf("직선의 끝점의 좌표 x, y를 입력하세요 : ");

scanf("%d %d", &c, &d);

length = GetLength(a, b, c, d);

printf("두 점 사이의 거리 : %f\n", length);

return 0;

}

double GetLength(int x1, int y1, int x2, int y2)

{

int dx = x2 - x1;

int dy = y2 - y1;

return sqrt(dx\*dx + dy\*dy);

}

double GetLength(int x1, int y1, int x2, int y2);

또는 double GetLength(int, int, int, int);

12. 다음 프로그램의 실행 결과를 쓰시오.

#include <stdio.h>

double dummy = 12.34;

int main(void)

{

double dummy = 56.78;

if( dummy > 20 )

{

double dummy = 16.25;

dummy++;

printf("dummy = %f\n", dummy);

}

return 0;

}

dummy = 17.250000

**CHAPTER 08 포인터 예제**

/\* Ex08\_01.c \*/

#include <stdio.h>

int main(void){

char \*pc;

int \*pi;

double \*pd;

printf("pc의 크기 : %d\n", sizeof(pc));

printf("pi의 크기 : %d\n", sizeof(pi));

printf("pd의 크기 : %d\n", sizeof(pd));

printf("char\* 의 크기 : %d\n", sizeof(char\*));

printf("short\* 의 크기 : %d\n", sizeof(short\*));

printf("int\* 의 크기 : %d\n", sizeof(int\*));

printf("float\* 의 크기 : %d\n", sizeof(float\*));

printf("double\*의 크기 : %d\n", sizeof(double\*));

return 0;

}

**<EX 08\_01 실행결과>**

**pc의 크기 : 4**

**pi의 크기 : 4**

**pd의 크기 : 4**

**char\* 의 크기 : 4**

**short\* 의 크기 : 4**

**int\* 의 크기 : 4**

**float\* 의 크기 : 4**

**double\*의 크기 : 4**

**Program ended with exit code: 0**

/\* Ex08\_02.c \*/

#include <stdio.h>

int main(void){

int x;

int \*p;

p = &x;

\*p = 10;

printf("\*p = %d\n", \*p);

printf("x = %d\n", x);

printf("p = %p\n", p);

printf("&x = %p\n", &x);

printf("&p = %p\n", &p);

return 0;

**<EX 08\_02 실행결과>**

**\*p = 10**

**x = 10**

**p = 0012FF7C**

**&x = 0012FF7C**

**&p = 0012FF78**

**Program ended with exit code: 0**

// Ex08\_04

#include <stdio.h>

int main(void) {

int arr[5] = {10, 20, 30, 40, 50};

int \*p = &arr[0];

int i;

for(i = 0; i < 5; i++, p++)

printf("%d\n", \*p);

return 0;

}

**<EX 08\_04 실행결과>**

**10**

**20**

**30**

**40**

**50**

**Program ended with exit code: 0**

// Ex08\_05

#include <stdio.h>

int main(void) {

int arr[5] = {10, 20, 30, 40, 50};

int i;

for(i = 0; i < 5; i++) {

printf("&arr[%d] = %p ", i, &arr[i]);

printf("arr+%d = %p\n", i, arr+i);

}

for(i = 0; i < 5; i++) {

printf("&arr[%d] = %d ", i, arr[i]);

printf("arr+%d = %d\n", i, \*(arr+i));

}

return 0;

}

**<EX 08\_05 실행결과>**

**&arr[0] = 0x7ffeefbff570 arr+0 = 0x7ffeefbff570**

**&arr[1] = 0x7ffeefbff574 arr+1 = 0x7ffeefbff574**

**&arr[2] = 0x7ffeefbff578 arr+2 = 0x7ffeefbff578**

**&arr[3] = 0x7ffeefbff57c arr+3 = 0x7ffeefbff57c**

**&arr[4] = 0x7ffeefbff580 arr+4 = 0x7ffeefbff580**

**&arr[0] = 10 arr+0 = 10**

**&arr[1] = 20 arr+1 = 20**

**&arr[2] = 30 arr+2 = 30**

**&arr[3] = 40 arr+3 = 40**

**&arr[4] = 50 arr+4 = 50**

**Program ended with exit code: 0**

// Ex08\_07

#include <stdio.h>

#include <string.h>

int main(void) {

char \*p = "abcde";

printf("p = %s\n", p);

printf("p = %p\n", p);

printf("p = %p\n", "abcde");

// p[0] = 'A';

// strcpy(p, "hello");

return 0;

}

**<EX 08\_07 실행결과>**

**p = abcde**

**p = 0x100000f5e**

**p = 0x100000f5e**

**Program ended with exit code: 0**

//Ex08\_08

#include <stdio.h>

#include <string.h>

int main(void) {

char \*p = "abcde";

printf("p = %s\n", p);

printf("p = %p\n", p);

printf("p = %p\n", "abcde");

p = "hello";

printf("p = %s\n", p);

printf("p = %p\n", p);

printf("p = %p\n", "hello");

return 0;

}

**<EX 08\_08 실행결과>**

**p = abcde**

**p = 0x100000f4e**

**p = 0x100000f4e**

**p = hello**

**p = 0x100000f64**

**p = 0x100000f64**

**Program ended with exit code: 0**

**CHAPTER 08 포인터 연습문제**

1. 포인터에 대한 설명 중 잘못된 것을 모두 고르시오.

① 포인터는 다른 변수를 가리키는 변수이다.

② 포인터 변수는 다른 변수의 이름를 저장한다.

③ 포인터 변수도 변수이므로 주소를 구할 수 있다.

④ 포인터 변수의 크기는 포인터가 가리키는 변수의 크기와 같다.

⑤ 포인터 변수가 가리키는 변수에 접근하려면 \* 연산자를 이용한다.

⑥ 변수의 주소를 구할 때는 & 연산자를 이용한다.

2. 다음 중 포인터의 선언 및 초기화가 잘못된 것을 모두 고르시오.

① int x;

int \*p1 = &x;

② short s;

int \*p2 = &s;

③ char str[20];

char \*p3 = str;

④ double d;

double \*p4 = d;

⑤ float f;

float \*p5 = NULL;

3. 포인터 사용에 대한 설명 중 잘못된 것은?

① 포인터 변수의 데이터형은 포인터 변수가 가리키는 변수의 데이터형과 서로 달라도 된다.

② 포인터를 초기화하지 않고 사용하는 것은 위험하다.

③ 간접 참조 연산자(\*)는 포인터 변수에만 사용할 수 있다.

④ 주소 구하기 연산자(&)는 상수나 수식에는 사용할 수 없다.

4. 널 포인터에 대한 설명 중 잘못된 것은?

① NULL이라는 매크로 상수가 라이브러리에 정의되어 있다.

② NULL 매크로 대신 0을 사용해도 된다.

③ 포인터가 가리키는 변수가 없을 때 널 포인터로 만든다.

④ 널 포인터에 대하여 간접 참조 연산자(\*)를 사용하면 메모리 0번지에 접근한다.

5. 다음과 같이 short\*형의 포인터 변수가 선언되어 있을 때, 포인터 연산의 결과를 구

하시오.

short arr[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

short \*p = arr; // p에 저장된 arr의 주소가 0x8000번지라고 가정하자.

(1) p + 1 0X8002

(2) p + 3 0X8006

(3) \*p 10

(4) \*(p + 2) 30

(5) p[5]; 60

6. 다음과 같이 선언된 포인터 변수에 대한 코드의 실행 결과 중 잘못된 것은?

double num = 12.34; // num의 주소는 0x5678번지

double \*p = &num; // p의 주소는 0x5670번지

코드 실행결과

① printf(“%lf”, \*p); 12.340000

② printf(“%p”, p); 00005678

③ printf(“%p”, &p); 00005674

④ printf(“%p”, p + 1); 00005678

⑤ printf(“%d”, sizeof(p)); 4

7. 다음과 같이 선언된 int형 변수와 int\*형의 포인터 변수에 대하여 잘못된 코드는?

int data = 10;

int \*p = &data;

① p = NULL;

② p = &(data + 1);

③ \*p = 100;

④ p = &data + 1;

8. 배열과 포인터에 대한 설명 중 잘못된 것은?

① 배열의 이름은 배열의 시작주소가 되므로 포인터인 것처럼 사용할 수 있다.

② arr[i]는 \*(arr+i)를 의미한다.

③ 포인터 변수가 배열 원소를 가리키고 있을 때, 포인터를 배열인 것처럼 사용할 수 있다.

④ 배열의 시작 주소는 변경할 수 있다.

9. 다음과 같이 선언된 배열과 포인터 변수에 대해서 여러 가지 수식 중 의미가 다른

하나는?

int x[10];

int \*p = x;

① \*(x + i)

② x[i]

③ \*(p + i)

④ &p[i]

10. 다음과 같이 선언된 배열과 포인터 변수에 대한 코드 중 컴파일 에러가 발생하는 것

은?

int x[10], y[10];

int \*p = x;

① p++; ② x++;

③ p = y; ④ p = &x[3];

11. 가리키는 변수가 없다는 의미로 0번지를 저장하는 포인터를 무엇이라고 하는지 쓰

시오.

널 포인터

12. 문자열 포인터에 대한 설명 중 잘못된 것은?

① 문자열의 주소를 저장하려면 char\*형을 사용한다.

② 문자 배열의 주소를 저장할 때는 char\*형을 사용한다.

③ 문자열 리터럴의 주소를 저장할 때는 const char\*형을 사용하는 것이 좋다.

④ const char\*형의 포인터 변수가 가리키는 문자열은 변경할 수 있다.

13. 다음과 같이 선언된 문자 배열과 포인터가 있을 때 컴파일 에러나 실행 에러가 발생

할 수 있는 문장을 모두 고르시오.

char str[] = "hello world";

char \*p1 = "good bye";

char \*p2 = str;

① p1 = “Hello”; ② p1[0] = ‘G’;

③ p2[0] = ‘H’; ④ strcpy(p1, “abcde”);

⑤ strcpy(p2, “abcde”); ⑥ str = “Hello”;

14. 다음 프로그램에서 컴파일 에러가 발생하는 곳을 모두 찾으시오.

#include <stdio.h>

#include <string.h>

int main(void)

{

char str1[20] = "Hello World";

char str2[20] = "Good Bye";

const char \*p1 = str1;

char \* const p2 = str1;

const char \* const p3 = str1;

p1[0] = 'A'; // ①

p1 = str2; // ②

p2[0] = 'A'; // ③

p2 = str2; // ④

p3[0] = 'A'; // ⑤

p3 = str2; // ⑥

return 0;

}

①, ④, ⑤, ⑥

15. 다음과 같이 선언된 포인터 변수 pi, pc, pd에 대해서 pi의 값이 0x12FF60번지이고,

pc가 0x12FF40 번지이고, pd가 0x12FF00번지일 때, 포인터 연산의 결과를 구하

시오.

int arr1[5] = {10, 20, 30, 40, 50};

int \*pi = arr1; // pi는 0x12FF60번지

char arr2[] = "abcdefg";

char \*pc = arr2; // pc는 0x12FF40번지

double arr3[5] = {1.1, 2.2, 3.3, 4.4, 5.5};

double \*pd = arr3; // pd는 0x12FF00번지

(1) pi + 2 0x12FF68

(2) \*(pi + 1) 20

(3) pc[3] d

(4) &arr3[3]-&arr3[1] 2

(5) &pd[1] 0x12FF08

16. 다음은 포인터로 배열에 접근해서 배열의 원소를 1 증가 시킨 후 출력하는 프로그

램이다. \_\_\_\_\_\_\_\_\_\_\_\_\_에 필요한 코드를 작성하시오. (단, 반드시 p를 이용해서

배열의 원소에 접근해야 한다.)

#include <stdio.h>

int main()

{

int arr[5] = {10, 20, 30, 40, 50};

int \*p = NULL;

int i;

p = arr;

for( i = 0 ; i < 5 ; i++ )

{

printf("%d ", );

p++;

}

printf("\n");

return 0;

}

++\*p 또는 ++(\*p) 또는 ++p[0]

17. 다음 프로그램의 실행 결과를 쓰시오.

#include <stdio.h>

int main()

{

char str[] = "abcdefghij";

char \*p = NULL;

p = str + 1;

p[0] = 'X';

p = str + 3;

p[1] = 'Y';

p = str + 5;

p[2] = 'Z';

printf("str = %s\n", str);

return 0;

}

str = aXcdYfgZij