20173853 이찬준

C프로그래밍 포트폴리오

과제 제출

목차

11장

12장

13장

11장 문자와 문자열

11장의 학습 목표는 다음과 같다.
1. 문자와 문자열의 기본적 이해
2. 문자와 문자열의 입출력 및 관련 함수
3. 여러 개의 문자열을 처리하는 방법

[1. 문자와 문자열]

◎ 문자와 문자열

문자	ex) char ch = 'A';	'작은따옴표' 사용
마지막에 \0가 저장되어야한다. = 문자열이 저장되는 배열크기는 저장될 문자 수 보다 1이 커야한다. ex) char ch = 'A'; char csharp[3]; csharp[0] = 'C'; csharp[1] = '#'; csharp[2] = '\0';		

문자열	ex) char c[] = "C language";	"큰따옴표" 사용
마지막에 \0가 저장되어야한다. = 문자열이 저장되는 배열크기는 저장될 문자 수 보다 1이 커야한다. ex) //문자 하나 하나 저장 시 마지막에 '\0' 문자 저장 char java[] = {'J', 'A', 'V', 'A', '\0'}; 마지막 '\0'을 빼면 출력 시 문제가 발생한다.		

▶ 배열 초기화시 배열크기는 지정하지 않는 것이 더 편리, 지정한다면 '\0'을 고려해 실제 문자 수 보다 1이 더 크게 배열크기를 지정해야함.

▶ 지정한 배열크기가 문자수 +1 보다 크면 나머지는 모두 '\0'으로 채워진다.

◎ '\0'에 의한 문자열 분리

```
char c[] = "C C++ Java";
c[5] = '\0';
printf("%s \n%s\n", c, (c+6);

'\0' 때문에 c[0]~c[4]와 c[6]~c[10]. 두 개의 문자열로 인식된다.
```

[출력결과]

```
C C++
Java
```

◎ 문자 입력에 쓰이는 함수들의 특징

함수	scanf()	getchar()	getche() _getche()	getch() _getch()
헤더파일	<stdio.h>		<conio.h>	
버퍼 이용	버퍼 이용함		버퍼 이용안함	
반응	Enter키를 눌러야 작동		문자 입력마다 반응	
입력 문자의 표사(echo)	누르면 바로 표시		누르면 바로 표시	표시 안 됨
입력문자 수정	가능		불가능	

◎ char()와 ch()의 차이점

	입력	출력
버퍼처리함수	getchar()	putchar()
문자입출력함수	getch()	putch()

◎ 문자열 입출력에 쓰이는 함수들

	함수		특징	
입력	scanf()	공백으로 구분되는 하나의 문자열 입력	다양한 입출력에 적합	
출력	puts()	공백으로 구분되는 하나의 문자열 출력		
입력	gets()	한행의 문자열 입력	처리 속도가 빠름	마지막 \n가 \0로 교체되어 저장
출력	puts()	한행의 문자열 출력		마지막 \0이 \n으로 교체되어 저장

[2. 문자열 관련 함수]

◎ 문자열 비교 함수 strcmp() strncmp()

	기본형		반환값
strcmp()	int strcmp(const char * s1, const char * s2);	같은 위치의 문자를 앞에서부터 **다를 때까지** 비교.	같으면 0 앞이 크면 양수 뒤가 크면 음수
strncmp( )	int strncmp(const char * s1, const char * s2, size_t maxn);	같은 위치의 문자를 앞에서부터 **n까지** 비교	
특징	▶ const가 붙으면 문자열을 수정할 수 없다. ▶ 사전(아스키코드) 순서로 비교		

◎ 문자열 복사 함수 strcpy() strncpy()

	기본형	반환값
strcpy()	char strcpy(char * dest, const char * source);	const 유무의 차이로 앞 문자열은 수정되지만 뒤 문자열은 수정될 수 없다.
strncpy()	char strncpy(char * dest, const char * source, size_t maxn);	size_t maxn(범위)가 안 써있으면 끝까지 복사
특징	▶ strcpy()과 strncpy()의 차이는 범위 차이	

◎ 문자열 연결 함수 strcat() strncat()

	기본형	반환값
strcat()	char strcat(char * dest, const char * source);	앞 문자열 dest에 뒤 문자열 source를 연결해 저장
strncat()	char strncpy(char * dest, const char * source, size_t maxn);	size_t maxn(범위)가 안 써있으면 끝까지 연결
특징	▶ strcpy()과 strncpy()의 차이는 범위 차이	

◎ 문자열 토큰 추출 함수 strtok()

	기본형	반환값
strtok()	char strtok(char * str, const char * delim);	앞 문자열 str에서 뒤 문자열 delim을 구성하는 구분자를 기준으로 토큰을 추출하여 반환하는 함수
특징	▶ str은 문자열 상수를 사용할 수 없고, 문자배열에 저장된 문자열을 사용해야 한다. ▶ 결과를 저장한 ptoken이 NULL이면 더 이상 분리할 토큰이 없는 경우다.	

◎ 문자열 길이와 위치 검색 함수 strlen()

	반환값
strlen()	Null문자를 제외한 문자열 길이를 반환
strlwr()	인자를 모두 소문자로 변환하여 반환
strupr()	인자를 모두 대소문자로 변환하여 반환

[3. 여러 문자열 처리]

◎ 문자 포인터 배열 방법과 이차원 문자 배열 방법

	기본형	특징

문자 포인터 배열	char *pa[] = {“JAVA”, “C#”, “C++”};
	//각각의 3개 문자열 출력
	printf(“%s ”, pa[0]); printf(“%s ”, pa[1]); printf(“%s\n”, pa[2]);
		▶하나의 문자 포인터가 하나의 문자열을 참조 -> 문자 포인터 배열은 여러 개의 문자열을 참조
		▶여러 개의 문자열을 처리하는데 유용

		장점 - 최적의 공간 사용 단점 - 수정 불가능

이차원 문자 배열	char ca[][5] = {“JAVA”, “C#”, “C++”};
	//각각의 3개 문자열 출력
	printf(“%s ”, ca[0]); printf(“%s ”, ca[1]); printf(“%s\n”, ca[2]);
		▶모든 열 수 동일하게 메모리에 할당
		▶여러 개의 문자열을 처리하는데 유용

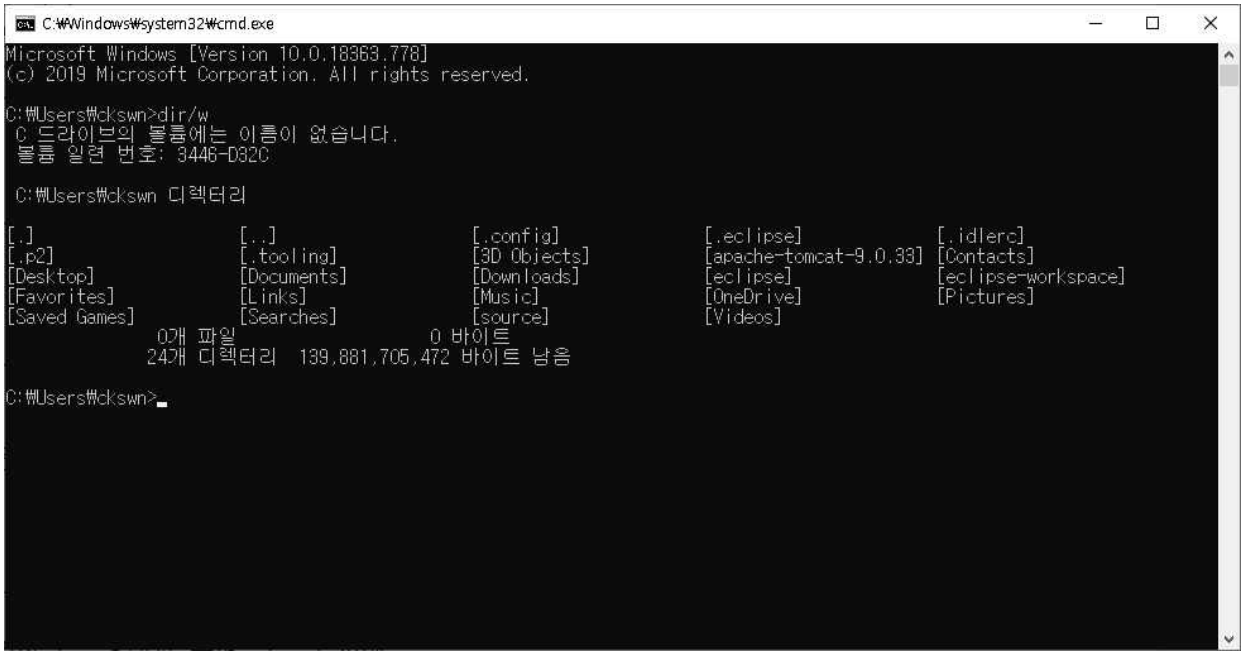
		장점 - 문자열을 수정가능 단점 - ‘\0’문자가 들어가 낭비되는 메모리 공간 발생 가능

◎ 명령행 인자의 필요성과 구현 방법의 이해

▷예시그림

▶ 명령행 인자(command line arguments)를 사용하는 방법: 명령행에서 입력하는 문자열을 프로그램으로 전달하는 방법

▶ 프로그램에서 명령행 인자를 받으려면 main() 함수에서 두 개의 인자 argc와 argv를 (int argc, char * argc[])로 기술해야 한다.



▶ 매개변수 argc: 명령행에서 입력한 문자열의 수. argc는 명령행에서 입력한 개수에 +1을 해줘야 한다. 이유는 자기 자신(프로그램)도 개수에 들어가기 때문

▶ argc[]: 명령행에서 입력한 문자열을 전달 받는 문자 포인터 배열

▶ 실행 프로그램이름(여기서는 dir)도 하나의 명령행 인자에 포함된다.

[실습예제 11-1 chararray]

```
#include <stdio.h>

int main(void)
{

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

	char java[] = { 'J', 'A', 'V', 'A' ,'\0'};
	printf("%s\n", java);

	char c[] = "C Language";
	printf("%s\n", c);

	char csharp[5] = "C#";
	printf("%s\n", csharp);

	printf("%c%c \n", csharp[0], csharp[1]);

	return 0;
}
```

```
A 65
JAVA
C language
C#
C#
```

[실습예제 11-2 charpointer]

```
#include <stdio.h>

int main(void)
{

	char* java = "java";
	printf("%s ", java);

	int i = 0;
	while (java[i]);
	printf("%c", java[i++]);
	printf(" ");

	i = 0;
	while (*(java + i) != '\0')
		printf("%c", *(java + i++));
	printf("\n");

	java[0] = 'J';

	return 0;
}
```

```
java java java
```

[실습예제 11-3 string]

```
#include <stdio.h>

int main(void)
{
	char c[] = "C C++ Java";
	printf("%s\n", c);
	c[5] = '\0';
	printf("%s\n %s\n", c, (c + 6));

	c[5] = ' ';
	char* p = c;
	while (*p)
		printf("%c", *p++);
	printf("\n");

	return 0;
}
```

```
C C++ Java
C C++
Java
C C++ Java
```

[실습예제 11-4 getche]

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

```
int main(void)
{
	char ch;

	printf("문자를 계속 입력하고 Enter를 누르면 >>\n");
	while ((ch = getchar()) != 'q')
		putchar(ch);

	printf("\n문자를 누를 때마다 두 번 출력 >>\n");
	while ((ch = _getche()) != 'q')
		_putch(ch);

	printf("\n문자를 누르면 한 번 출력 >>\n");
	while ((ch = _getch()) != 'q')
		_putch(ch);
	printf("\n");

	return 0;
}
```

```
문자를 계속 입력하고 Enter를 누르면 >>
java
java
python
python
q

문자를 누를 때마다 두 번 출력 >>
jjaavvaaq
문자를 누르면 한 번 출력 >>
java
```

[실습예제 11-5 stringput]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>


int main(void)
{
	char name[20], dept[30];

	printf("%s", "학과 입력>>");
	scanf("%s", dept);
	printf("%s", "이름 입력>>");
	scanf("%s", name);
	printf("출력: %10s %10s\n", dept, name);

	return 0;
}
```

```
학과 입력 >> 컴퓨터 정보 공학과
이름 입력 >> 김미림
출력: 컴퓨터 정보 공학과 김미림
```

[실습예제 11-6 gets]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main(void)
{
	char line[101];

	printf("입력을 종료하려면 새로운 행에서 (ctrl+Z)를 누르십시오.\n");
	while (gets(line))
		puts(line);
	printf("\n");

	while (gets_s(line, 101))
		puts(line);
	printf("\n");

	return 0;
}
```

```
입력을 종료하려면 새로운 행에서 (ctrl + Z)를 누르십시오.
문자열 처리를 배우고 있습니다.
문자열 처리를 배우고 있습니다.
^z

gets()의 사용도 마찬가지입니다.
gets()의 사용도 마찬가지입니다.
^z
```

[실습예제 11-7 memfun]

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

int main(void)
{
	char src[50] = "http://www.visualstudio.com";
	char dst[50];
	printf("문자배열 src = %s\n", src);
	printf("문자열크기 strlen(src) = %d\n", strlen(src));
	memcpy(dst, src, strlen(src) + 1);
	printf("문자배열 dst = %s\n", dst);
	memcpy(src, "안녕하세요!", strlen("안녕하세요") + 1);
	printf("문자배열 src = %s\n", src);

	char ch = ':';
	char* ret;
	ret = memchr(dst, ch, strlen(dst));
	printf("문자 %c 뒤에는 문자열 %s 이 있다.\n", ch, ret);

	return 0;
}
```

```
문자배열 src = http://www.visualstudio.com
문자열크기 strlen(src) = 28
문자배열 dst = http://www.visualstudio.com
문자배열 src = 안녕하세요!
문자 : 뒤에는 문자열 : //www.visualstudio.com 이 있다.
```

[실습예제 11-8 strcmp]

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

int main(void)
{
	char* s1 = "java";
	char* s2 = "java";
	printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));

	s1 = "java";
	s2 = "jav";
	printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
	s1 = "jav";
	s2 = "java";
	printf("strcmp(%s, %s) = %d\n", s1, s2, strcmp(s1, s2));
	printf("strcmp(%s, %s, %d) = %d\n", s1, s2, 3,
strncmp(s1, s2, 3));

	return 0;
}
```

```
strcmp(java, java) = 0
strcmp(java, jav) = 1
strcmp(jav, java) = -1
strcmp(jav, java, 3) = 0
```

[실습예제 11-9 strcpy]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include<string.h>

int main(void)
{
	char dest[80] = "Java";
	char sourcep[80] = "C is language";

	printf("%s\n", strcpy(dest, source));
	printf("%s\n", strcpy(dest, "C#", 2));
	printf("%s\n", strcpy(dest, "C#", 3));

	return 0;
}
```

```
C is a language.
C# is a language.
C#
```

[실습예제 11-10 strcat]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
```

```
int main(void)
{
	char dest[80] = "C";

	printf("%s\n", strcat(dest, " is "));
	printf("%s\n", strcat(dest, "a java", 2));

	printf("%s\n", strcat(dest, "procedural "));
	printf("%s\n", strcat(dest, "language."));

	return 0;

}
```

```
C is
C is a
C is a procedural
C is a procedural language.
```

[실습예제 11-11 strtok]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
	char str1[] = "C and C++\t language are best!";
	char* delimiter = " , \t!";
```

```
        printf("문자열 \"%s\" 을 >> \n", str1);
        printf("구분자 [%s]를 이용하여 토큰을 추출 >>\n",
delimiter);
        char* ptoken = strtok(str1, delimiter);

        while (ptoken != NULL)
        {
                printf("%s\n", ptoken);
                ptoken = strtok(NULL, delimiter);

        }
        return 0;
}
```

```
문자열 “C and C++     language are best!”을 >>
구분자[ ,          !]를 이용하여 토큰을 추출 >>
C
and
C++
language
are
best
```

[실습예제 11-12 strfun]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
```

```
int main(void)
{
	char str[] = "JAVA 2017 go c#";
	printf("%d\n", strlen("java"));
	printf("%s, ", _strlwr(str));
	printf("%s\n ", _strupr(str));

	printf("%s, ", strstr(str, "VA"));

	printf("%s\n", strchr(str, 'A'));

	return 0;
}
```

```
4
java 2017 go c#, JAVA 2017 GO C#
VA 2017 GO C#, AVA 2017 GO C#
```

[실습예제 11-13 strarray]

```
#include <stdio.h>

int main(void)
{
	char* pa[] = { "JAVA", "C#", "C++" };
	char ca[][5] = { "JAVA", "C#", "C++" };

	printf("%s ", pa[0]); printf("%s ", pa[1]); printf("%s\n",
```

```
pa[2]);
        printf("%s ", ca[0]); printf("%s ", ca[1]); printf("%s\n", ca[2]);

        printf("%c %c %c\n", pa[0][1], pa[1][1], pa[2][1]);
        printf("%c %c %c\n", ca[0][1], ca[1][1], ca[2][1]);

        return 0;
}
```

JAVA C# C++

JAVA C# C++

A # +

A # +

[실습예제 11-14 commandarg]

```
#include <stdio.h>

int main(int argc, char *argv[])
{
        int i = 0;

        printf("실행 명령행 인자(command line arguments) >>\n");
        printf("argc = %d\n", argc);
        for (i = 0; i < argc; i++)
                printf("argc[%d] = %s\n", i, argv[i]);

        return 0;
```

```
}
```

```
실행 명령행 인자(command line arguments) >>
argc = 4
argc[0] = G:\[2016 C]\Ch11\Debug\Prj14.exe\
argc[1] = C#
argc[2] = C++
argc[3] = Java
```

[Lab 11-1 lineprint]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>


int main()
{
	char s[100];
	gets(s);

	char* p = s;
	while (*p)
		printf("%c", *p++);
	printf("\n");

	return 0;
}
```

```
char *p = s;
```

```
while (*p)
    printf("%c", *p++);
```

[Lab 11-2 strreverse]

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

int reverse(char str[]);

int main(void)
{
        char s[50];
        memcpy(s, "C Programming!", strlen("C Programming!",
strlen("C Programming!" + 1);
        printf("%s\n", s);

        reverse(s);
        printf("%s\n", s);

        return 0;
}

void reverse(char str[])
{
        for (int i = 0, j = strlen(str) - 1; i < j; i++, j--)
        {
                char c = str[i];
```

```
            str[i] = str[j];
            str[j] = c;
        }
}
```

```
#include <string.h>
memcpy(s, “C Programming!”, strlen(“C Programming!”)+1);
str[i] = str[j];
```

[Lab 11-3 strprocess]

```
#include <stdio.h>

int main(void)
{
        char str1[] = "JAVA";
        char str2[] = "C#";
        char str3[] = "C++";

        char* pstr[] = { str1, str2, str3 };

        printf("%s ", pstr[0]);
        printf("%s ", pstr[1]);
        printf("%s ", pstr[2]);

        printf("%c %c %c\n", str1[0], str2[1], str3[2]);
        printf("%c %c %c\n", pstr[0][1], pstr[1][1], pstr[2][1]);
```

```
        return 0;
}
```

```
char *pstr[] = { str1, str2, str3 };
printf("%s ", pstr[1]);
    printf("%c %c %c\n", pstr[0][1], pstr[1][1], pstr[2][1];
```

12장 변수 유효 범위

12장의 학습 목표는 다음과 같다.

1. 변수 유효범위의 기본적 이해

2. 정적 변수와 레지스터 변수의 이해

[1. 지역변수와 전역변수]

◎ 지역변수와 전역변수

	선언위치	참조 가능 범위	특징
지역(地域)변수	함수or블록 내부	그 지역 내	▶ 선언 후 초기화 하지 않으면 쓰레기 값이 저장되므로 주의
전역(全域)변수	함수 외부	프로젝트 내 모든 함수와 블록 참조 가능	▶ 자동으로 초기값이 자료형에 맞는 0으로 지정 ▶ 키워드 extern을 사용하여 이미 존재하는 전역변수의 유효범위 확장 가능

▷예시그림

[2. 정적변수와 레지스터 변수/ 3. 메모리 영역과 변수 이용]

◎ 기억부류의 이해

기억부류 종류	전역	지역
auto	X	O
register	X	O
static	O	O
extern	O	X
point: 변수는 4가지의 기억 부류에 따라 할당되는 메모리 영역이 결정되고 메모리의 할당과 제거시기가 결정됨.		

◎ 기억부류에 따른 변수선언

	기본형
기억부류 변수 선언	auto int n; register double yield; static double data = 5.85; int age; extern int input;

◎ 각 기억부류의 특징

register	▶ CPU 내부에 있는 기억장소 이므로 일밤 메모리보다 빠르게 참조 된다. ▶ 일반 메모리에 할당되는 변수가아니므,로 주소연산자&를 사용할 수 없다.
static	▶ 정적변수는 최초 생성 이후 메모리에서 제거 되지않으므로 지속적으로 저장값을 유지 및 수정가능 ▶ 정적변수는 초기값을 지정하지 않으면 자동으로 자료형에 따라 0이나 '\0'또는 NULL 값이 저장

	▶ 정적변수의초기화는 상수로 단 한 번. 더 이상 초기화 되지 않는다. ▶ 지역변수라고 하더라도 함수가 종료 돼도 static지역변수가 없어지는 것이 아니다.
extern	▶ 전역변수의 유효범위를 확장 할 수 있다. ▶ 프로그램이 크고 복잡하면 전역변수의 사용은 원하지 않는 전역변수의 수정과 같은 부작용의 위험성이 항상 존재하므로 최대한 쓰지 않는 것이 좋다.

◎ 정적변수의 초기화

정적변수의 초기화	#include <stdio.h>
	
	int a = 1;
	static s = 1; //전역변수에 쓰는 static
	
	int main(void)
	{
	int data = 10;
	static value = 10; //지역변수에 쓰는 static
	
	return 0;
	}

[실습예제 12-1 localvar]

```
#include <stdio.h>

void sub(int param);

int main(void)
{
        auto int n = 10;
        printf("%d\n", n);

        for (int m = 0, sum = 0; m<3; m++)
        {
                sum += m;
                printf("\t%d %d\n", m, sum);
        }

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

        sub(20);

        return 0;
}

void sub(int param)
{
        auto int local = 100;
        printf("\t%d %d\n", param, local);
}
```

```
10
                0 0
```

```
                1 1
                2 3
10
                20 100
```

[실습예제 12-2 globalvar]

```
#include <stdio.h>

double getArea(double);
double getCircum(double);

double PI = 3.14;
int gi;

int main(void)
{
        double r = 5.87;
        const double PI = 3.141592;

        printf("면적: %.2f\n", getArea(r));
        printf("둘레1: %.2f\n", 2 * PI*r);
        printf("둘레2: %.2f\n", getCircum(r));
        printf("PI: %f\n", PI);
        printf("gi: %d\n", gi);

        return 0;
}
```

```
double getArea(double r)
{
	return r * r * PI;
}
```

```
면적: 108.19
둘레1: 36.88
둘레2: 3 6 8 6
PI: 3.141592
gi: 0
```

[실습예제 12-3 circumference]

```
extern double PI;

double getCircum(double r)
{
	return 2 * r * PI;
}
```

[실습예제 12-4 registervar]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main()
{
	register int sum = 0;

	int max;
	printf("양의 정수 입력 >> ");
	scanf("%d", &max);

	for (register int count = 1; count <= max; count++)
		sum += count;

	printf("합: %d\n", sum);

	return 0;
}
```

```
양의 정수 입력 >> 8
합: 36
```

[실습예제 12-5 staticlocal]

```
#include <stdio.h>


void increment(void);

int main()
{
	for (int count = 0; count < 3; count++)
		increment();
```

```
}

void increment(void)
{
        static int sindex = 1;
        auto int aindex = 1;
        printf("정적 지역변수 sindex: %2d, \t", sindex++);
        printf("자동 지역변수 aindex: %2d, \t", aindex++);
}
```

```
정적 지역변수 sindex: 1,        자동 지역변수 aindex: 1
정적 지역변수 sindex: 2,        자동 지역변수 aindex: 1
정적 지역변수 sindex: 3,        자동 지역변수 aindex: 1
```

[실습예제 12-6 staticgvar]

```
#include <stdio.h>

static int svar;
int gvar;

void increment();
void testglobal();

int maun(void)
{
        for (int count = 1; count <= 5; count++)
                increment();
        printf("함수 increment()가 총 %번 호출되었습니다.\n",
```

```
svar);

        testglobal();
        printf("전역 변수: %d\n", gvar);
}

void increment()
{
        svar++;
}
```

```
함수 increment()가 총 5번 호출되었습니다.
전역 변수: 10
```

[실습예제 12-7 gfunc]

```
void teststatic()
{
}

void testglobal()
{
        extern gvar;
        gvar = 10;
}
```

[실습예제 12-8 storageclass]

```
#include <stdio.h>

void infunction(void);
void outfunction(void);

int global = 10;
static int sglobal = 20;

int main(void)
{
        auto int x = 100;

        printf("%d, %d, %d\n", global, sglobal, x);
        infunction(); outfunction();
        infunction(); outfunction();
        infunction(); outfunction();

        return 0;
}

void infunction(void)
{
        auto int fa = 1;
        static int fs;

        printf("\t%d, %d ,%d ,%d\n", ++global, ++sglobal, fa,
++fs);
}
```

```
10, 20, 100
```

```
        11, 21, 1, 1
                12
        13, 22, 1, 2
                14
        15, 23, 1, 3
                16
16, 23, 100
```

[실습예제 12-9 out]

```
#include <stdio.h>

void outfunction()
{
        extern int global, sglobal;

        printf("\t\t%d\n", ++global);
}
```

[Lab 12-1 fibonacci]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int count;
```

```
void fibonacci(int prev_number, int number);

void main() {

	auto prev_number = 0, number = 1;

	printf("피보나츠를 몇 개 구할까요?(3 이상) >> ");

	scanf("%d", &count);
	if (count <= 2)
		return 0;
	printf("1 ");
	fobonacci(prev_number, number);
	printf("\n");
}
void fibonacci(int prev_number, int number)
{

	static int i = 1;

	while (i++ < count)
	{
		int next_num = prev_number + number;
		prev_number = number;
			number = next_num;
		printf("%d ", next_num);
		fibonacci(prev_number, number);
	}
}
```

```
static int I = 1;
```

```
int next_num = prev_number + number;
fibonacci(prev_number, number);
```

[Lab 12-2 static]

```
#include <stdio.h>

void process();

void main()
{

	process();
	process();
	process();

	return 0;
}

void process()
{

	static int sx;

	int x = 1;

	printf("%d %d\n", x, sx);
```

```
        x += 3;
        sx += x + 3;
}
```

```
1 0
1 7
1 14
```

[Lab 12-3 bank]

```
#include <stdio.h>

int total = 10000;

void save(int);
void withdraw(int);

int main(void)
{
        printf(" 입금액      출금액 총입금액      총출금액      잔      고
\n");

printf("==========================================================
=\n");
        printf("%46d\n", total);
        save(50000);
        withdraw(30000);
        save(60000);
        withdraw(20000);
```

```
printf("================================================
=\n");

        return 0;
}

void save(int money)
{
        static int amount;
        total += money;
        amount += money;
        printf("%7d %17d %20d\n", money, amount, total);
}

void withdraw(int money)
{
        static int amount;
        total -= money;
        amount += money;
        printf("%15d %20d %9d\n", money, amount, total);
}
```

```
int total = 10000;
save(50000);
static int amount;
total -=money;
```

13장 구조체(struct)와 공용체(union)

13장의 학습 목표는 다음과 같다.

1. 구조체와 공용체의 기본적 이해

2. 자료형 재정의를 위한 typedef에 대한 이해

3. 구조체 포인터와 배열의 이해

[1. 구조체와 공용체]

◎ 구조체란?

연관성 있는 서로 다른 개별적인 자료형의 변수들을 하나의 단위로 묶은 새로운 자료형

	구조체 변수 선언 기본형		이름 없는 구조체의 변수 선언

구조체	struct lecture
	{
	char name[20];
	int credit;
	int hour;
	};
		struct lecture
		datastructure;
			struct
			{
			char name[20];
			int credit;
			int hour;
			} youraccount;

◎ 구조체의 활용

	구조체의 활용	
구조체	struct date { int year; int month; int day;	struct account { struct date open; char name[12]; int actnum;

	};	double balance; }; struct account me = { {2012,3, 9}, “홍길동”, 1001, 300000 };

◎ 구조체의 동등비교

if (one == bae) printf(“내용이 같은 구조체입니다.\n”); //오류	한 번에 비교는 안 됨
if (one.snum == bae.snum) printf(“학번이 %d로 동일합니다.\n”, one.snum); //오류	하나씩 비교해야 함
if (one.snum == bae.snum && !strcmp(pne.name, bae.name) && !strcmp(one.dept, bae.dept)) printf(“내용이 같은 구조체입니다.\n”);	

◎ 문자열을 처리하기 위한 포인터 char *와 배열 char[]

char *dept;	char name[12];
상수로 문자열을 처리하기 용이	저장을 하고 수정할 가능성이 있을 때 사용
dept = “컴퓨터정보공학과”;	name = “나한국”; //오류
strcpy(dept, “컴퓨터정보공학과”); //오류	strcpy(name, “배상문”);
scanf(“%s”, dept); //오류	scanf(“%s”, name);

◎ 공용체란?

동일한 저장소에 여러 자료형을 저장하는 방법

공용체	구조체 변수 선언 기본형	공용체 변수의 크기는 멤버중 가장 큰 자료형의크기로 정해진다. (여기서는 flaot) 마지막에 저장된 하나의 멤버 자료값만 유효하다.

	union share
	{
	int count;
	float value;
	};
	
	union share a;	

◎ 멤버 접근 연산자 .와 변수 크기

접근연산자 .를 통해 멤버를 참조 할 수 있다.

접근 연산자 .	접근 연산자 사용 기본형

	구조체변수이름.멤버
	mine.actnum = 1002;
	mine.balance = 3000000;

▶ 변수 mine의 크기는 sizeof(mine)으로 알아볼 수 있다. ▶ 실제 구조체의 크기는 멤버의 크기의 합보다 크거나 같다.	

[2. 자료형 재정의]

◎ typedef란?

이미 사용되는 자료유형을 다른 새로운 자료형 이름으로 재정의할 수 있도록 하는 키워드

	typedef 구문 기본형	구조체 정의와 typedef를 함께이용한 자료형의 정의

자료형 재정의 typedef	struct date
	{
	int year;
	int month;
	int date;
	};
	
	typedef struct date date;
		typedef struct
		{
		char title[30];
		char company[30];
		char kinds[30];
		date release;
		} software;
		//software는 변수가 아닌 새로운 자료형

[3. 구조체와 공용체의 포인터와 배열]

◎ 구조체 포인터와 배열을 활용할 수 있다.

구조체의 주소를 저장하는 포인터의 선언과 활용

	typedef 구문 기본형

구조체 포인터 변수 선언	struct lecture
	{
	char name[20];
	int type;
	int credit;
	int hours;
	};
	typedef struct lecture lecture;
	lecture *p;

◎ 포인터 변수의 구조체 멤버 접근 연산자 ->

p -> name //name은 필드명	연산자 ->와 .	우선순위: 1위 결합성: 좌->우,
	연산자 *	우선순위: 2위 결합성: 우->좌

◎ 구조체 변수와 구조체 포인터 변수를 이용한 멤버의 참조

p->name (*p).name	포인터 p가 가리키는 구조체의 멤버 name
*p.name	*(p.name)이고 p가 포인터이므로 p.name은 문법오류 발생

◎ 구조체 배열의 선언과 활용방법

동일한 구조체 변수가 여러 개 필요하면 다른 배열과 같이 구조체 배열을 선언 하여 사용할 수 있다.

구조체 배열의 선언 기본형	
lecture c[] = { {“인간과 사회”, 0, 2, 2}, {“경제학개론”, 1, 3, 3}, {“자료구조”, 2, 3, 3}, };	외부 중괄호: 배열 초기화 내부 중괄호: 배열원소인 구조체 초기화

[실습예제 13-1 structbasic]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

struct account
{
	char name[12];
	int actnum;
	double balance;
};

int main(void)
{
	struct account mine = { "홍길동", 1001, 30000 };
	struct account yours;

	strcpy(yours.name, "이동원");
	yours.actnum = 1002;
	yours.balance = 500000;

	printf("구조체크기: %d\n", sizeof(mine));
	prinft("%s %d %.2f\n", mine.name, mine.actnum, mine.balance);
	prinft("%s %d %.2f\n", yours.name, yours.actnum, yours.balance);

	return 0;
}
```

구조체크기: 24

```
홍길동 1001 300000.00
이동원 1002 500000.00
```

[실습예제 13-2 nestdestruct]

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

struct date
{
	int year;
	int month;
	int day;
};

struct account
{
	struct date open;
	char name[12];
	int actnum;
	double balance;
};

int main(void)
{
	struct account me = { { 2018, 3, 9 }, "홍길동", 1001, 300000 };
```

```
	printf("구조체크기: %d\n", sizeof(me));
	printf("[%d, %d, %d]\n", me.open.year, me.open.month, me.open.day);
	printf("%s %d %.2f\n", me.name, me.actnum, me.balance);
}
```

```
구조체크기: 40
[2018, 3, 9]
홍길동 1001 300000.00
```

[실습예제 13-3 structstudent]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
{
	struct student
	{
		int snum;
		char* dept;
		char name[12];
	};
	struct student hong = { 201800001, "컴퓨터정보공학과", "홍길동" };
	struct student na = { 201800002 };
	struct student bae = { 201800003 };
```

```
	scanf("%s", na.name);

	na.dept = "컴퓨터정보공학과";
	bae.dept = "기계공학과";
	memcpy(bae.name, "배상문", 7);
	strcpy(bae.name, "배상문");
	strcpy_s(bae.name, 7, "배상문");

	pruntf("[%d, %s, %s]\n", hong.snum, hong.dept, hong.name);
	pruntf("[%d, %s, %s]\n", na.snum, na.dept, na.name);
	pruntf("[%d, %s, %s]\n", bae.snum, bae.dept, bae.name);



	struct student one;
	one = bae;
	if (one.snum == bae.snum)
		printf("학번이 %d으로 동일합니다.\n", one.snum);

	if (one.snum == bae.snum && !strcmp(one.name, bae.name) && !strcmp(one.dept, bae.dept))
		printf("내용이 같은 구조체입니다.\n");

	return 0;
}
```

```
나한국
[201800001, 컴퓨터정보공학과, 홍길동]
[201800002, 컴퓨터정보공학과, 나한국]
[201800003, 기계공학과, 배상문]
학번이 201800003(으)로 동일합니다.
```

내용이 같은 구조체입니다.

[실습예제 13-4 union]

```
#include <stdio.h>

union data
{
	char ch;
	int cnt;
	double real;
} data1;

int main(void)
{
	union data data2 = { 'A' };
	union data data3 = data2;

	printf("%d %d\n", sizeof(union data), sizeof(data3));

	data1.ch = 'a';
	printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
	data1.cnt = 100;
	printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);
	data1.real = 3.156759;
	printf("%c %d %f\n", data1.ch, data1.cnt, data1.real);

	return 0;
```

```
}
```

```
8 8
a 97 0.000000
d 100 0.000000
N -590162866 3.156759
```

[실습예제 13-5 typedef]

```
#include <stdio.h>

typedef unsigned int budget;

int main(void)
{
	budget year = 24500000;

	typedef int profit;

	profit month = 4600000;

	printf("올 예산은 %d, 이달의 이익은 %d 입니다.\n", year, month);

	return 0;
}

void test(void)
{
```

```
        budget year = 24500000;

}
```

올 예산은 24500000, 이달의 이이긍ㄴ 4600000입니다.

[실습예제 13-6 typedefstruct]

```
#include <stdio.h>

struct date
{
        int year;
        int month;
        int day;
};

typedef struct date date;

int main(void)
{
        typedef struct
        {
                char title[30];
                char company[30];
                char kinds[30];
                date release;
        }software;
```

```
        software vs = { "비주얼스튜디오 커뮤니티", "MS", "통합개발
환경", {2018, 8, 29} };

        printf("제품명: %s\n", vs.title);
        printf("회사: %s\n", vs.company);
        printf("종류: %s\n", vs.kinds);
        printf("출시일:    %d.    %d.    %d\n",    vs.release.year,
vs.release.month, vs.release.day);

        return 0;
}
```

```
제품명: 비주얼스튜디오 커뮤니티
회사 :MS
종류 : 통합개발환경
출시일: 2018. 8. 29
```

[실습예제 13-7 structpointer]

```
#include <stdio.h>

struct lecture
{
        char name[20];
        int type;
        int credit;
        int hours;
};
typedef struct lecture lecture;
```

```
char *head[] = { "강좌명", "강좌구분", "학점", "시수" };

char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };

int main(void)
{
	lecture os = { "운영체제", 2, 3, 3 };
	lecture c = { "c프로그래밍", 3, 3, 4 };
	lecture *p = &os;

	printf("구조체크기: %d, 포인터크기: %d\n\n", sizeof(os), sizeof(p));
	printf("%10s %12s %6s %6s\n", head[0], head[1], head[2], head[3]);
	printf("%12s %10s %5d %5d\n", p -> name, lectype[p -> type], p -> credit, p -> hours);

	p = &c;
	printf("%12s %10s %5d %5d\n", (*p).name, lectype[(*p).type], (*p).credit, (*p).hours);
	printf("%12c %10s %5d %5d\n", *c.name, lectype[c.type], c.credit, c.hours);

	return 0;
}
```

```
구조체크기: 32, 포인터크기: 4

    강좌명       강좌구분     학점     시수
    운영체제     전공필수       3        3
C프로그래밍      전공선택       3        4
```

[실습예제 13-8 unionpointer]

```
#include <stdio.h>

int main(void)
{
	union data
	{
		char ch;
		int cnt;
		double real;
	};

	typedef union data udata;

	udata value, * p;

	p = &value;
	p->ch = 'a';
	printf("%c %c\n", p->ch, (*p).ch);
	p->cnt = 100;
	printf("%d ", p->cnt);
	p->real = 3.14;
	printf("%.2f \n", p->real);

	return 0;
}
```

```
a a
100 3.14
```

[실습예제 13-9 stuctarray]

```
#include <stdio.h>

struct lecture
{
	char name[20];
	int type;
	int credit;
	int hours;
};
typedef struct lecture lecture;

char* lectype[] = { "교양", "일반선택", "전공필수", "전공선택" };
char* head[] = { "강좌명", "강좌구분", "학점", "시수" };

int main(void)
{
	lecture course[] = { {"인간과 사회", 0, 2, 2 },
	{ "경제학개론", 1, 3, 3 },
	{ "자료구조", 2, 3, 3 },
	{ "모바일 프로그래밍", 2, 3, 4 },
	{ "고급 C프로그래밍", 3, 3, 4 }};

	int arysize = sizeof(course) / sizeof(course[0]);

	printf("배열크기: %d\n\n", arysize);
	printf("%12s %12s %6s %6s\n", head[0], head[1],
head[2], head[3]);

printf("==================================================
```

```
==========\n");
        for (int i = 0; i < arysize; i++)
                printf("%16s %10s %5d %5d\n", course[i].name,
lectype[course[i].type], course[i].credit, course[i].hours);

        return 0;
}
```

```
배열크기: 5

          강좌명          강좌구분       학점         시수
==================================================
         인간과 사회           교양          2            2
          경제학개론       일반선택          3            3
            자료구조       전공필수          3            3
    모바일프로그래밍       전공필수          3            4
   고급 C프로그래밍       전공선택          3            4
```

```
[Lab 13-1 bank]

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

struct position
{
        double latitude;
        double longitude;
};

int main(void)
```

```
{
	struct city
	{
		char* name;
		struct position place;
	};
	struct city seoul, newyork;

	seoul.name = "서울";
	seoul.place.latitude = 37.33;
	seoul.place.longitude = 126.58;

	newyork.name = "뉴욕";
	newyork.place.latitude = 40.8;
	newyork.place.longitude = 73.9;
}
```

```
struct city seoul, newyork;
seoul.name = “서울”;
seoul.place.latitude = 37.33;
seoul.place.longitude = 126.58;
newyork.name = “뉴욕”;
newyork.place.latitude = 40.8;
newyork.place.longitude = 73.9;
```

[Lab 13-2 structcity]

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

```
int main(void)
{

	typedef struct movie
	{
		char* title;
		int attendance;
	}movie;
	movie assassination;

	assassination.title = "암살";
	assassination.attendance = 12700000;

	printf("[%s] 관객수: %d\n", assassination.title, assassination.attendance);

	return 0;

}
```

```
} movie;
movie assassination;
```

[Lab 13-3 structmovie]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main(void)
```

```
{
	typedef struct movie
	{
		char* title;
		int attendance;
		char director[20];
	} movie;

	movie box[3] = {
		{ "명량", 17613000, "김한민"},
		{ "국제시장", 14257000, "윤제균"},
		{ "베테랑", 13383000} };

	strcpy(box[2].director, "류승완");

	printf("	제목	감독	관객수\n");
	printf("================================\n");
	for (int i = 0; i < 3; i++)
		printf("[%8s %6s %d\n]",
			box[i].title, box[i].director, box[i].attendance);

	return 0;
}
```

```
box[] = {
strcpy(box[2].director, “류승완”);
    box[i].title, box[i].director, box[i].attendance);
```