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

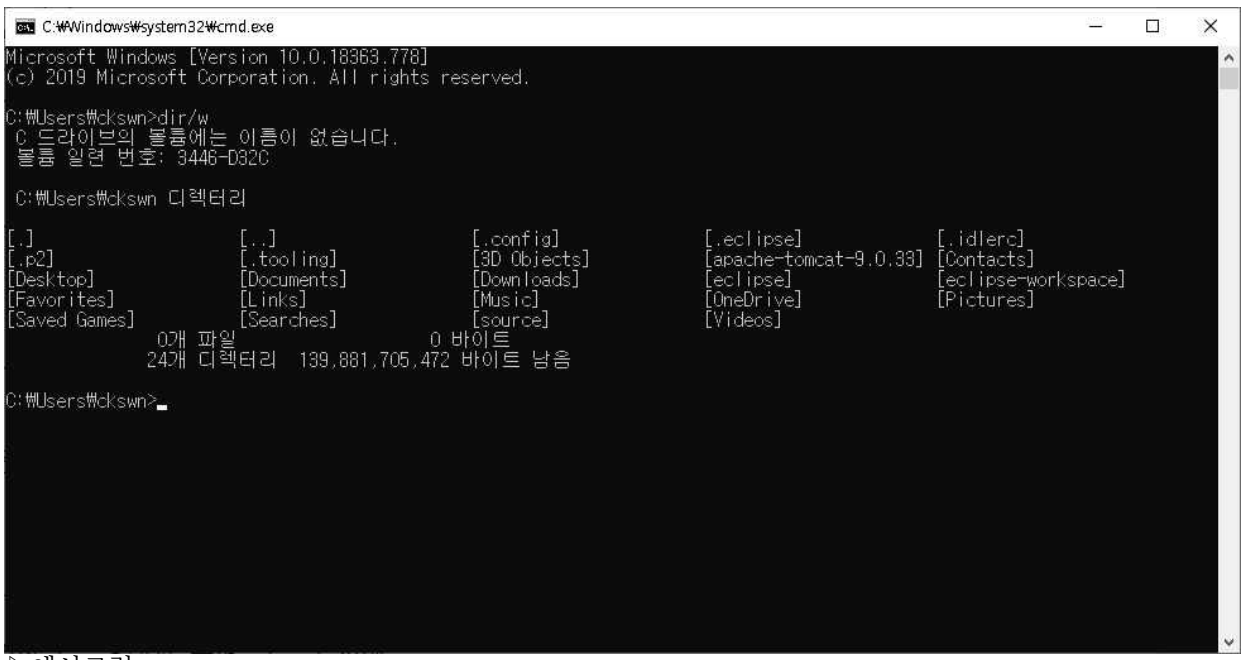
문자 포인터 배열	char *pa[] = {“JAVA”, “C#”, “C++”};
	//각각의 3개 문자열 출력
	printf(“%s ”, pa[0]); printf(“%s ”, pa[1]); printf(“%s\n”, pa[2]);
		▶하나의 문자 포인터가 하나의 문자열을 참조 -> 문자 포인터 배열은 여러 개의 문자열을 참조
		▶여러 개의 문자열을 처리하는데 유용

		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 입

력하는 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.

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 참조된다. ▶ 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 없어지

	는 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;
	}

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

<table>
<tr><td rowspan="2">구조체</td><td colspan="2">구조체 변수 선언 기본형</td><td>이름 없는 구조체의 변수 선언</td></tr>
<tr><td>struct lecture
{
char name[20];
int credit;
int hour;
};</td><td>struct lecture
datastructure;</td><td>struct
{
char name[20];
int credit;
int hour;
} youraccount;</td></tr>
</table>

◎ 구조체의 활용

<table>
<tr><td rowspan="2">구조체</td><td colspan="2">구조체의 활용</td></tr>
<tr><td>struct date
{
int year;
int month;
int day;
};</td><td>struct account
{
struct date open;
char name[12];
int actnum;
double balance;
};
struct account me =
{ {2012,3, 9}, “홍길동”, 1001, 300000 };</td></tr>
</table>

◎ 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

	구조체 변수 선언 기본형	

공용체	union share
	{
	int count;
	float value;
	};
	
	union share a;
		공용체 변수의 크기는 멤버중 가장 큰 자료형의크기로 정해진다. (여기서는 flaot)
		
		마지막에 저장된 하나의 멤버 자료값만 유효하다.

◎ 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	포인터 p가 가리키는 구조체의 멤버 name
(*p).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