Emotion C언어 보고서

10707 고태건

목차

변수

형식 이름	바이트	기타 이름	값의 범위
int	4	signed	–2,147,483,648 ~ 2,147,483,647
unsigned int	4	unsigned	0 ~ 4,294,967,295
__int8	1	char	-128 ~ 127
unsigned __int8	1	unsigned char	0 ~ 255
__int16	2	short, short int, signed short int	–32,768 ~ 32,767
unsigned __int16	2	unsigned short, unsigned short int	0 ~ 65,535
__int32	4	signed, signed int, int	–2,147,483,648 ~ 2,147,483,647
unsigned __int32	4	unsigned, unsigned int	0 ~ 4,294,967,295
__int64	8	long long, signed long long	–9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
unsigned __int64	8	unsigned long long	0 ~ 18,446,744,073,709,551,615
bool	1	없음	false 또는 true
char	1	없음	-기본적으로 128 ~ 127 /J를 사용하여 컴파일된 경우 0~255

signed) char	**1**	**없음**	**-128 ~ 127**
unsigned char	**1**	**없음**	**0 ~ 255**
short	**2**	**short int, signed short int**	**–32,768 ~ 32,767**
unsigned short	**2**	**unsigned short int**	**0 ~ 65,535**
long	**4**	**long int, signed long int**	**–2,147,483,648 ~ 2,147,483,647**
unsigned long	**4**	**unsigned long int**	**0 ~ 4,294,967,295**
long long	**8**	**없음 (하지만 같음 __int64)**	**–9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807**
unsigned long long	**8**	없음 (하지만 같음 **unsigned __int64**)	**0 ~ 18,446,744,073,709,551,615**
enum	**varies**	**없음**	
float	**4**	**없음**	**3.4E+/-38(7개의 자릿수)**
double	**8**	**없음**	**1.7E+/-308(15개의 자릿수)**
long double	동일 **double**	**없음**	동일 **double**
wchar_t	**2**	**__wchar_t**	**0 ~ 65,535**

연산자

대입 연산자 =

= 변수에 어떤 값을 대입 할 때 사용

산술 연산자 +, -, *, /, %, ++, —

+ 더하기
- 빼기
* 곱하기
/ 나누기
% 나머지
++ 1 증가
— 1 감소

관계 연산자 <, >, <=, >=, ==, !=

> 오른쪽이 왼쪽보다 크면 1 작으면 0
< 오른쪽이 왼쪽보다 크면 0 작으면 1
>= 오른쪽이 왼쪽보다 크거나 같으면 1 작으면 0
<= 오른쪽이 왼쪽보다 크거나 같으면 0 작으면 1
== 오른쪽과 왼쪽보다 같으면 1 다르면 0
!= 오른쪽과 왼쪽보다 같으면 0 다르면 1

논리 연산자 &&, ||, !

&& AND 양쪽이 모두 참이면 참
|| OR 양쪽 중 하나만 참이면 참
! NOT 참이면 거짓, 거짓이면 참

할당 연산자 +=, -=, *=, /=, %= 등…

+= 자신의 값에 오른쪽을 더한다
-= 자신의 값에 오른쪽을 뺀다
*= 자신의 값에 오른쪽을 곱한다
/= 자신의 값에 오른쪽을 나눈다
%= 자신의 값에 오른쪽을 나눈 나머지를 구한다

삼항 연산자 ?

조건 ? 참일때의 값 : 거짓일때의 값;

비트 연산자 &, |, ~, ^, <<, >>

- & 비트 단위 AND 연산
- | 비트 단위 OR 연산
- ~ 비트 단위 NOT 연산
- ^ 비트 단위 XOR 연산
- << 왼쪽으로 비트 이동
- >> 오른쪽으로 비트 이동

입/출력

#stdio.h

standar**d** **i**nput **o**utput => **stdio**

printf(“content”); // 출력
scanf(“content”); // 입력

#conio.h

console **i**nput **o**utput => **conio**

getchar();
getch();
getchar();
getche();
putch();

	getchar();	getch();	getchar();	getche();	putch();
버퍼 사용	O	X	X	O	X
화면 표시	O	X	O		
입력 종료	\n	\n	\r		

조건문

```
if()
	if(조건1){ // 조건1에 부합할때
		실행;
	}
else
	if(조건){ // 조건1에 부합할때
		실행;
	} else { // 조건1이 아닐때
		실행;
	}
else if()
	if(조건1){
		실행;
	} else if(조건2) { // 조건1이 아니고 조건2일때
		실행;
	} else { //조건1이 아니고 조건2도 아닐때
		실행;
	}
switch() case
	switch(변수){
		case 값1: // 변수가 값1일때
			실행;
		case 값2: // 변수가 값2일때
			실행;
		case 값3: // 변수가 값3일때
			실행;
		default: // 모든 case문이 해당되지 않을때
			실행;
	}
```

반복문

for()

for(초기식; 조건식; 변화식){
실행;
}

초기식: 반복문을 시작할 때 초기식입니다. 반복에 사용할 변수는 초기식 부분에서 선언해도 되고, for 반복문 바깥에서 선언해도 됩니다.
조건식: 반복될 조건입니다. 조건식이 참이면 계속 반복하며, 거짓이 되면 반복문을 끝냅니다.
변화식: 반복문이 한 번 실행될 때마다 수행할 식입니다.

while()

while(조건식){
실행;
}

조건식: 반복될 조건입니다. 조건식이 참이면 계속 반복하며, 거짓이 되면 반복문을 끝냅니다.

do-while()

do{
실행;
}while(조건식)

조건식: 반복될 조건입니다. 조건식이 참이면 계속 반복하며, 거짓이 되면 반복문을 끝냅니다.

break;

break; 는 조건문을 끝낼때 사용합니다.

continue;

continue; 는 뒤의 코드를 무시하고 다시 반목문의 조건식으로 제어를 옮깁니다.

함수

함수의 형태

```
반환값 함수명(인자){
    실행;
}
```

함수의 요소

반환형: 반환형이란 함수가 실행을 마치고 돌려주는 결과값의 자료형입니다. 반환문은 지금 있는 함수를 빠져나가는 역할을 하는데, 반환형이 있을 경우는 return 뒤에 변수나 상수 등을 붙여서 값을 반환할 수 있게 하는 제어문입니다. 함수는 무조건 한 개의 반환형을 가지고, 선언할때에는 그 자료형을 써줍니다. 만약 반환이 필요 없을 때에는 void 자료형을 써줍니다.

함수명: 함수명은 그 함수의 고유한 이름입니다.

인자: 선언시에는 자료형과 변수명으로 이루어지며, 실제 사용할 때에는 해당 자료형에 맞는 변수나 상수가 들어가게 됩니다.

int형 함수

```
int 함수명(인자){
    실행;
    return 반환형;
}
```

void형 함수

```
void 함수명(인자){
    실행;
}
```

재귀함수

재귀함수는 어떤 함수 내에서 자기 자신을 또다시 호출하는 방식을 말합니다.

재귀호출은 보통 위의 예처럼 작은 단위의 작업이 반복되는 곳에 많이 쓰입니다. 재귀호출을 이용한 함수를 작성할때에는 무한 루프에 빠지지 않도록 함수가 잘 종료될 수 있게 하여야 합니다.

포인터

정의

```
자료형 *포인터이름;
포인터 = &변수;
```

포인터는 메모리 주소를 저장하는 변수이다.

포인터 변수를 선언할 때는 자료형 뒤에 * 를 붙입니다. *의 위치에 따른 차이는 없으며 모두 같은 뜻입니다.

구조체

정의

```
struct 구조체이름{
    자료형 멤버이름;
}
```

구조체는 정의만 해서는 사용을 할 수가 없습니다. 따라서 구조체도 변수로 선언해서 사용합니다.

선언

```
struct 구조체이름 변수이름;
```

구조체에서 멤버를 불러올때는 다음과 같이 사용한다.

```
구조체이름.변수이름;
```

연결 리스트

형태 (노드가 두개인 리스트)

```
HEAD -> NODE 1 -> NODE 2 -> NULL
```

코드

```
struct NODE {   // 연결 리스트의 노드 구조체
    struct NODE *next;    // 다음 노드의 주소를 저장할 포인터
    int data;             // 데이터를 저장할 멤버
};
```

요소

- 머리 노드: 연결 리스트의 기준점이며 헤드(head)라고도 부릅니다. 머리 노드는 첫 번째 노드를 가리키는 용도이므로 데이터를 저장하지 않습니다.
- 노드: 단일 연결 리스트에서 데이터가 저장되는 실제 노드입니다.

파일 입출력

파일 입출력은 fopen() 함수를 사용한다.

형태

FILE * fopen(const char *, const char *);
첫번째 인자 : 처리할 파일 명
두번째 인자 : 파일 처리 종류 지정 (모드)

파일 처리 종류 (모드)

- r 읽기 모드 / 파일이 없을 경우 에러 발생
- w 쓰기 모드 / 파일이 없을 경우 새로 만들고, 파일이 존재하면 내용을 삭제하고 처음부터 기록
- a 추가 쓰기 모드 / 파일이 없을 경우 새로 만들고, 파일이 존재하면 뒤에부터 이어서 기록