

c 언어 보고서

2021210088 허지혜

2단원. 상수와 데이터 출력

1) c 프로그램의 구조와 데이터 출력 방법

- c 프로그램은 main 함수로 시작한다.
- //은 주석문이고 /**/는 여러 줄을 한꺼번에 주석 처리하는 주석문이다.
- printf 함수는 데이터를 화면에 출력할 때 사용한다.
- 제어 문자를 문자열 안에 포함시키면 그 기능에 따라 출력 형태가 바뀐다.

☞ 제어 문자 : printf가 인식하는 '문자' 중 출력 방식에 영향을 주는 문자를 의미한다. 백슬래시(\)와 함께 이용된다. 제어 문자의 종류는 다음과 같다.

제어 문자	의미	기능
\n	new line	줄을 바꾼다.
\t	tab	출력 위치를 다음 탭 위치로 옮긴다.
\r	carriage return	출력 위치를 줄의 맨 앞으로 옮긴다.
\b	backspace	출력 위치를 한 칸 왼쪽으로 옮긴다.
\a	alret 경보	벨 소리를 낸다.

- printf 함수로 숫자를 출력할 때는 정수 %d, 실수 %f 변환 문자를 사용한다.
이러한 문자를 서식 문자라고 한다. 서식 문자의 종류는 다음과 같다.

서식 문자	설명
%d, %i	10진수 정수(양,음수 둘 다 표현)
%f, %lf	10진수 실수(양,음수 둘 다 표현)
%s	문자열
%c	한 개의 문자

요기서 %lf는 소수점 이하 6자리까지 실수 출력을 하고 %.1lf는 소수점 이하 첫째 자리에서 출력한다.(둘째 자리에서 반올림)

2) 상수와 데이터 표현 방법

- 상수(constant)는 프로그램이 실행되는 동안 값이 고정되어 변경할 수 없는 메모리 공간을 의미하며 값은 바꿀 수 없으나 변수처럼 정의해서 사용할 수 있다. 10은 정수 상수, 10.0은 실수 상수, 'a'는 문자 상수(단일 문자), "a"는 문자열 상수이다. 논리 상수로는 true 또는 0이 아닌 수를 나타내는 참과 false 또는 0을 나타내는 거짓이 있다.
- 8진수는 %o, 16진수 소문자 출력은 %x, 대문자 출력은 %X를 사용한다.
- 3.14e-5처럼 소수점 앞 0이 아닌 유효 숫자 한자리를 이용해 지수 형태로 바꾼 것을 정규화 표기법이라고 한다.
- 정수 상수의 양수는 4바이트 크기의 2진수로, 음수는 2의 보수로 컴파일된다.
- 실수는 IEEE 754 표준의 double 형식에 따라 번역되며, 첫 비트는 부호 비트, 이후 11개 비트는 지수부, 나머지 52비트는 소수부를 나타낸다.

예시)

```

chapter2 (전역 범위) main0
#include <stdio.h>

main()
{
    printf("5*4-10 = 10\n");
    int a = 5, b = 4, c = 10;
    printf("%d * %d - %d = %d\n", a, b, c, a * b - c);
    a = a * b - c;
    printf("a 메모리 [%d] = %d\n", &a, a);
}

```

Microsoft Visual Studio 디버그 콘솔

```

5*4-10 = 10
5 * 4 - 10 = 10
a 메모리 [13629332] = 10

```

출처 : <https://opentutorials.org/module/3921/23508>

&a는 뒤에서 나올 주소연산자를 쓴 것이다. 변수가 어디 저장되어 있고 값이 무엇인지 나타내주는 코드이다.

제일 위에 #include <stdio.h>는 헤더 파일이라고 부른다. main()보다 먼저 작성하고 전처리 구문이라고 부르며 유사한 종류의 라이브러리 함수들이 포함되어 있다. 또한 헤더 파일에는 사용하는 명령들의 내용이 미리 기록되어 있다.

3단원. 변수와 데이터 입력

1) 변수

- 변수 선언으로 메모리에 저장 공간을 확보하며 대입 연산자로 변수값을 초기화하거나 저장한다. 초기화하지 않은 변수에는 쓰레기 값이 들어 있다.
- 변수의 형태를 자료형이라고 하며 기본적으로 정수형과 실수형으로 나눈다.

자료형	byte	값의 저장 범위	출력 변환 문자
char	1	-128~127	%c 또는 %d
short	2	-32768~32767	%d
int	4	-2147483648~2147483647	%d
long	4	-2147483648~2147483647	%ld
long long	8	-2 ⁶³ ~ 2 ⁶³ -1	%lld
unsigned char	1	0~255	%u
unsigned short	2	0~65535	%u
unsigned int	4	0~4294967295	%u
unsigned long	4	0~4294967295	%lu
unsigned long long	8	0~2 ⁶⁴ -1	%llu

- 변수에 const를 사용하면 상수처럼 사용할 수 있다.
- 예약어는 컴파일러와 약속된 단어로 c언어에서 미리 사용하겠다고 지정되어 있는 언어이다. 식별자는 사용자가 만들어낸 단어로 변수나 함수 등의 이름을 지정할 때 사용한다. 따라서 예약어는 식별자로 사용할 수 없다.

구분	예약어
자료형	char double enum float int long signed struct union unsigned void
제어문	break case continue default do else for goto if return switch while
기억클래스	auto extern register static
기타	const sizeof typedef volatile

2) 데이터 입력

- 키보드로 데이터를 입력할 때는 scanf 함수를 사용하여 변수 앞에 &를 사용한다.
- 둘 이상의 데이터를 입력할 때는 space bar, tab, enter로 구분한다.
- 문자열 입력은 char 배열을 이용하며 배열명 앞에 & 기호를 사용하지 않는다.

예시)

```
#define _CRT_SECURE_NO_WARNINGS // scanf 보안 경고로 인한 컴파일
#include <stdio.h>

int main()
{
    int num1;

    printf("정수를 입력하세요: ");
    scanf("%d", &num1); // 표준 입력을 받아서 변수에 저장

    printf("%d\n", num1); // 변수의 내용을 출력

    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
정수를 입력하세요: 30
30
```

출처 : <https://dojang.io/mod/page/view.php?id=79>

#define_CRT_SECURE_NO_WARNINGS를 넣지 않고 scanf를 그냥 사용하면 컴파일 에러가 난다. scanf_s 함수를 사용할 수도 있으나, c언어 표준 함수는 아니고 vs c++ 전용 함수이다.

4단원. 연산자

: 연산을 수행하는 기호를 연산자라고 하고 연산자 사이에 있는 값을 피연산자라고 한다.

1) 산술 연산자, 관계 연산자, 논리 연산자

- 대입 연산자는 오른쪽 수식의 값을 왼쪽 변수에 저장하며, 두 값이 같으리지를 확인할 때는 관계 연산자 ==을 사용한다.
- 산술 연산자 중 나누기 연산자(/)로 정수를 나누면 몫이 계산되며, 나머지는 나머지 연산자(%)로 연산한다.
- a++처럼 증감 연산자를 후위 표기하면 변수의 값을 사용하고 난 후에 증가시킨다.
- 논리연산자는 &&, ::, !가 있으며 논리 연산의 결과는 1 또는 0이 된다.

연산식	결과값
a > b	a가 b보다 크면 참

a >= b	a가 b보다 크거나 같으면 참
a < b	a가 b보다 작으면 참
a <= b	a가 b보다 작거나 같으면 참
a == b	a와 b가 같으면 참
a != b	a와 b가 다르면 참

관계 연산자

연산식	논리관계	결과값
a && b	논리곱(AND)	a,b 모두 참이면 참
a :: b	논리합(OR)	a,b 하나라도 참이면 참
!a	논리부정(NOT)	a가 거짓이면 참

논리 연산자

2) 그 외 유용한 연산자

- 형 변환 연산자는 피연산자의 값을 잠깐 원하는 형태로 바꾸나 변수의 형태는 바뀌지 않는다.
- sizeof 연산자는 괄호와 함께 사용하지만 함수는 아니다.
- 복합대입 연산자의 우선순위는 대입 연산자와 같다.
- 비트 연산자는 비트 단위로 연산하며 비트 논리 연산자와 비트 이동 연산자가 있다. 비트 논리 연산자 : &&, ::, ^ , 비트 이동 연산자 : >>, <<
- 연산자 우선 순위와 연산 방향을 살펴보면 다음과 같다.

단항 연산자 > 이항 연산자 > 삼항 연산자

산술 연산자 > 비트 이동 연산자 > 관계 연산자 > 논리 연산자

5단원. 선택문(if, switch ~ case)

1) if문

- 조건에 따라 실행 문장을 선택해야 할 때 선택문을 사용한다.
- 조건식을 만족할 때 실행할 실행문은 중괄호로 묶어준다.
- if문은 한 가지의 선택을 고민할 때 사용한다.
- if ~ else문은 둘 중에 하나를 고를 때 사용한다.
- if ~ else if ~ else문은 세 가지 이상에서 하나를 고를 때 사용한다.

구분 형식	
if (조건식) 실행문;	조건식이 참일 때 실행문 실행 거짓이면 아무것도 실행하지 않음
if (조건식) 실행문1; else 실행문2;	조건식이 참이면 실행문 1 실행 조건식이 거짓이면 실행문 2 실행
if (조건식1) 실행문1; else if (조건식2) 실행문2; else 실행문3;	조건식1이 참이면 실행문 1 실행 조건식1이 거짓이고 조건식2가 참이면 실행문 2 실행 모든 조건식이 거짓이면 실행문 3 실행

제어문은 크게 선택문, 반복문, 분기문으로 나뉜다.

선택문 : if, switch ~ case

반복문 : while for, do ~ while

분기문 : break, continue, goto, return (분기문은 조건문과 반복문에 중간에서 주어진 조건의 흐름을 바꿀 수 있는 구문이다.)

2) if문 활용과 switch ~ case 문

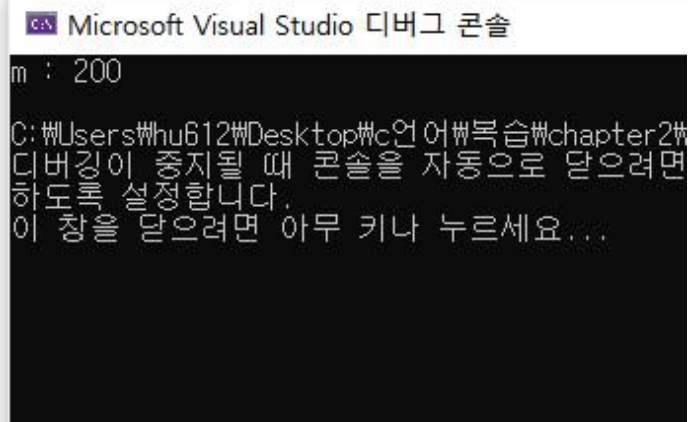
- 선행 조건이 꼭 필요한 경우 if문 중첩을 한다. if문 중첩이란 if문 안에 실행문으로 if문을 사용하는 것을 말한다. 이때 조건이 조금씩 다르다면 switch ~ case 문이 더 편리하다.
- switch ~ case문은 정수 값으로 실행할 문장을 결정한다.
- break를 생략할 때는 자세한 설명과 함께 제한적으로 사용해야 한다.
- default의 위치는 블록 안 어디에 와도 상관 없으나 마지막에 두어 예외 상황을 처리한다.

예시)

```
#define _CRT_SECURE_NO_WARNINGS // scanf 보안 경고로 인한 컴파일 에러
#include <stdio.h>

int main(void)
{
    int rank = 2, m = 0;

    switch (rank)
    {
        case 1:
            m = 300;
            break;
        case 2:
            m = 200;
            break;
        case 3:
            m = 100;
            break;
        default:
            m = 10;
            break;
    }
    printf("m : %d\n", m);
}
```



6단원. 반복문

1) while문, for문, do ~ while문

- while문은 반복 문장을 실행하기 전에 반복 조건을 먼저 검사한다.
- for문은 반복 횟수가 정해진 경우 사용하면 편리하다.
- do ~ while문은 반복 문장을 실행한 후에 반복 조건을 검사한다.

반복문 형식	실행 방식
while (조건식) { 실행문; }	조건식이 참인 동안 실행문을 반복한다. 최초 조건식이 거짓이면 실행문은 한 번도 실행되지 않는다.
for (초기식; 조건식; 증감식) { 실행문; }	초기식은 최초 한 번 실행한다. 조건식을 검사하여 참이면 실행문 -> 증감식 -> 조건식을 반복한다.

<pre> } do { 실행문; } while (조건식); </pre>	<p>실행문을 수행한 후에 조건을 검사한다. 조건식이 참인 동안 실행문을 반복한다. 실행문은 조건식과 관계없이 최소한 한번 실행된다.</p>
---	--

2) 반복문 활용

- 중첩 반복문은 반복문의 실행문으로 반복문을 사용한다.
- break와 continue를 사용하면 반복문의 실행 방식을 바꿀 수 있다.

중첩 반복문 예시	<pre> for (i=0;i<10;i++) { for (j=0;j<10;j++) { 반복할 문장; } } </pre>	I-for문이 10번 반복되고 j-for문이 10번 반복되므로 반복한 문장은 100번 반복된다.
분기문 사용 예시	<pre> while (1) { if (조건식1) break; if (조건식2) continue; 반복할 문장; } </pre>	조건식1이 참이면 반복문을 끝낸다. 조건식2가 참이면 반복할 문장을 건너뛰고 처음부터 다시 반복한다.

예시)

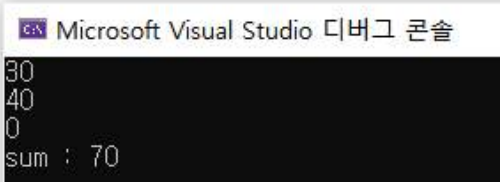

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main(void)
{
    int num;
    int sum = 0;
    do {
        scanf("%d", &num);
        sum += num;
    } while (num != 0);
    printf("sum : %d\n", sum);

    return 0;
}

```



Microsoft Visual Studio 디버그 콘솔

```

30
40
0
sum : 70

```

7단원. 함수

1) 함수의 작성과 사용

- 함수 선언을 하면 함수를 만들지 않고도 함수의 형태를 미리 알릴 수 있다.
- 함수 정의는 원하는 기능의 함수를 직접 만드는 것이다.
- 함수 호출은 만든 함수를 사용할 때 사용한다.
- retur은 함수를 실행한 다음 값을 반환할 때 사용하는 제어문이다.

구분	예	설명
함수 선언	int sum(int a, int b);	함수의 원형을 알린다.
함수 정의	<pre> int sum(int a, int b); { return a+b; } </pre>	함수를 만들고 안에 기능을 구현한다.
함수 호출	sum(10, 20);	함수를 사용한다. 함수에 필요한 값을 인수로 준다.

2) 여러 가지 함수 유형

- 처리할 데이터를 스스로 입력하는 함수에는 매개변수가 없어도 된다.
- 전달받은 데이터를 화면에 출력하는 함수는 반환형을 쓰지 않아도 된다.
- 같은 내용을 단지 화면에 출력하는 함수는 매개변수와 반환값을 둘 다 쓰지 않아도 된다.
- 매개변수와 반환값이 없을 때 빈 공간은 void를 적어준다.
- 재귀호출 함수는 자기 자신을 다시 호출한다.

매개변수가 없는 경우는 호출할 때 인수 없이 괄호만 사용한다.

반환형이 없는 경우는 반환할 때 return문을 쓰지 않거나 return 문만 사용한다.

반환형과 매개변수 모두 없는 경우는 두가지 특징을 포함한다.

재귀함수 호출 안에는 재귀호출을 멈추는 조건이 있어야 한다.

예시)

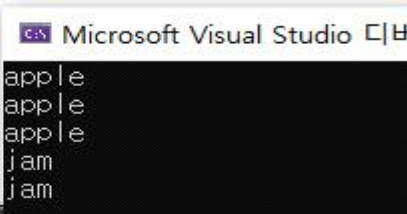
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

void fruit(int count);

int main(void)
{
    fruit(1);

    return 0;
}

void fruit(int count)
{
    printf("apple\n");
    if (count == 3) return;
    fruit(count + 1);
    printf("jam\n");
    return 0;
}
```



Microsoft Visual Studio 디버깅 콘솔 출력:

```
apple
apple
apple
jam
jam
jam
```

문제가 검색되었습니다

요기 예시는 책에서 나왔던 예시인데, fruit라는 함수 정의 안에 fruit가 또 들어가 있으므로 재귀함수이다. fruit(1)로 함수를 한 번 호출하여 apple이 출력되고 fruit(count+1)이라서 다시 fruit 함수로 돌아가야 하는데 이때 밑에 있는 jam을 한 번 출력해준다. 따라서 apple은 세 번, jam은 두 번이 출력된다.

8단원. 배열

1) 배열의 선언과 사용

- 배열을 선언하면 많은 변수를 한 번에 선언하는 효과를 볼 수 있다. 배열을 선언할 때는 변수 이름 뒤에 대괄호를 붙인 뒤 크기를 설정한다.
- 배열을 초기화할 때는 중괄호를 사용한다. 배열을 초기화할 때는 배열의 크기를 생략할 수 있다.
- 배열은 주로 반복문으로 처리한다.
- 배열 전체의 크기를 구할 때 sizeof 연산자를 사용한다.

2) 문자를 저장하는 배열

- char형 배열은 문자열을 저장하는 변수의 역할을 하는 것으로 문자열을 직접 초기화할 수 있다.
- char형 배열에 문자열을 저장할 때는 대입 연산자 대신 strcpy 함수를 사용한다.
- char형 배열에 문자열을 입출력할 때는 scanf, gets, printf, puts등의 함수를 사용한다.

예시) <https://jeak.tistory.com/33?category=832913>

5명의 학생의 성적을 입력 받고 출력시켜보세요.

10 1번째 학생의 성적은 10점입니다.

20 2번째 학생의 성적은 20점입니다.

30 3번째 학생의 성적은 30점입니다.

40 4번째 학생의 성적은 40점입니다.

50 5번째 학생의 성적은 50점입니다.

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

```
int main(void)
{
    int num1[5];
    printf("학생의 성적 : ");
    scanf("%d", &num1[0]);
    scanf("%d", &num1[1]);
    scanf("%d", &num1[2]);
    scanf("%d", &num1[3]);
    scanf("%d", &num1[4]);

    int i;

    for (i = 0; i < 5; i++)
    {
        printf("%d 번째 학생의 성적은 %d 점 입니다.\n", i+1, num1[i]);
    }
}
```

Microsoft Visual Studio 디버그 콘솔

```
학생의 성적 : 10
20
30
40
50
1 번째 학생의 성적은 10 점 입니다.
2 번째 학생의 성적은 20 점 입니다.
3 번째 학생의 성적은 30 점 입니다.
4 번째 학생의 성적은 40 점 입니다.
5 번째 학생의 성적은 50 점 입니다.
```