

12. 파일입출력

Autumn 2019

Today

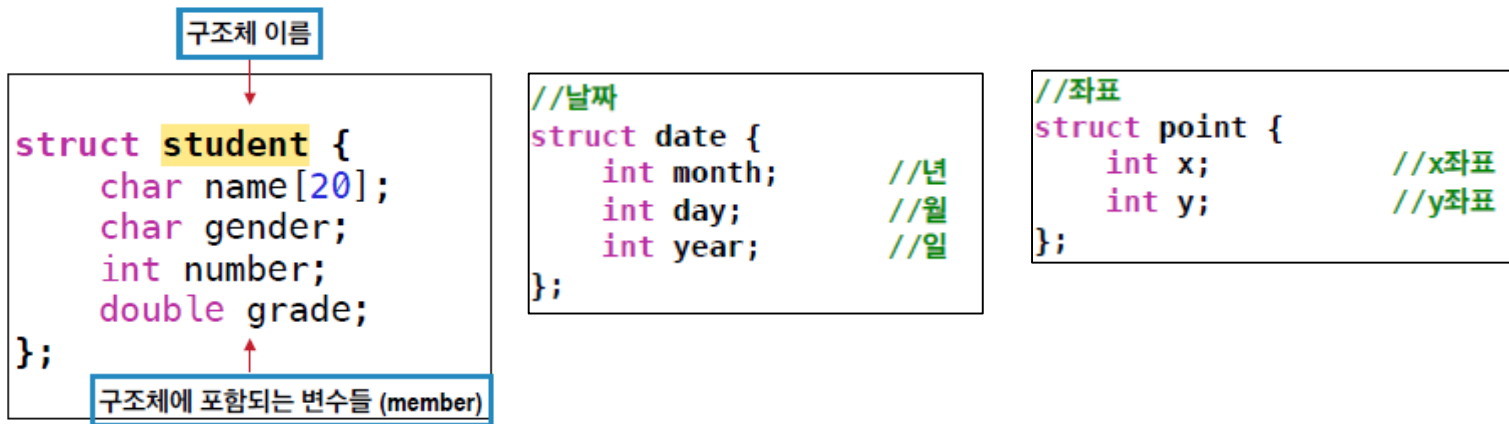
1. 지난 시간 복습 - 전처리문과 구조체
2. 메모리 할당하기
3. 파일입출력
4. 실습

[복습] 전처리문 (preprocessor)

- 컴파일하기 전에 미리 처리되는 문장
- 소스코드 시작부분에 위치하며, #으로 시작
- 종류:
#include, #define, #ifdef, #ifndef, #endif 등 다수

[복습] 구조체 (struct)

- 서로 다른 변수들을 하나로 묶어주는 기능
- 혹은 비슷한 기능을 하는 변수끼리 묶어주고 싶을 때,



[구조체 정의 형식]

- 구조체배열
- 구조체포인터

Today

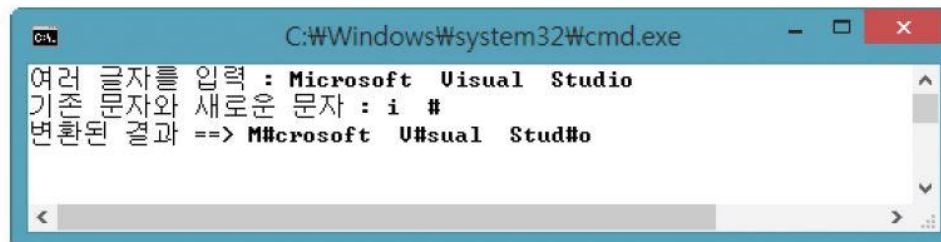
1. 지난 시간 복습 – 전처리문과 구조체
2. 메모리 할당하기
3. 파일입출력
4. 실습

[실습] 문자열 바꾸기 (1/2)

- 개수를 나타내는 변수 N 을 전처리기로 정의하고 초기값은 100으로 설정하기
- 문자열을 받아들일 char* 형 배열 N개 원소를 담도록 전역변수로 생성하기
- 바꿀글자와 기존글자를 입력받을 char 변수 두 개를 전역변수로 생성하기
- puts 와 gets 함수를 이용해서 여러 글자를 입력받기
- puts 와 scanf함수를 이용해서 기존글자와 바꿀글자 입력받기
- 입력받은 여러글자들 중 입력받은 글자로 변경하기

예제 설명 문자열을 입력받고 그 문자열에서 변환할 기존 문자와 새로운 문자를 각각 입력받은 뒤 변환된 문자열을 반환하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
여러 글자를 입력 : Microsoft Visual Studio
기존 문자와 새로운 문자 : i #
변환된 결과 ==> M#icrosoft V#sual Stud#o
```

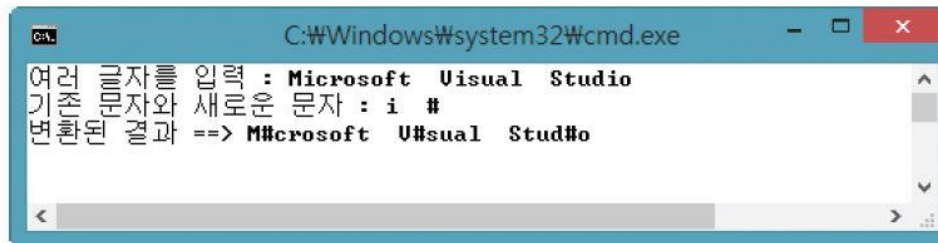
[실습] 문자열 바꾸기 (2/2)

2. 위 예제에서 바꿀문자를 비교해서 바꾸는 부분만 함수로 작성하시오. (함수명: `changeChar`, 반환값: 없음, 함수매개변수: 없음)
3. 전역에 선언된 배열과 문자 두 개 변수를 `main` 함수 안에서 선언하고, 함수를 수정한다. (함수명: `changeChar`, 반환값: `char*`, 함수매개변수: `char [], char, char`)
4. 매개변수 세 개를 구조체로 만들고 함수는 구조체를 매개변수로 받도록 수정한다.

==== 위 1~4은 모두 같은 결과를 나타내는 서로 다른 표현법이다.

예제 설명 문자열을 입력받고 그 문자열에서 변환할 기존 문자와 새로운 문자를 각각 입력받은 뒤 변환된 문자열을 반환하는 프로그램이다.

실행 결과



```
C:\Windows\system32\cmd.exe
여러 글자를 입력 : Microsoft Visual Studio
기존 문자와 새로운 문자 : i #
변환된 결과 ==> M#icrosoft U#sual Stud#o
```

메모리 동적관리

- 위 실습에서의 문제점?
- 사용자가 몇 글자를 넣을지 정해져 있지 않음
- 임의로 공간을 100개로 지정하고 있음 → 메모리낭비
- 해결책
 1. 메모리를 미리 잡아두지 않고, 사용할 공간의 크기를 사용자에게 물어봄
 2. 사용자가 답변한 크기만큼의 메모리를 확보
 - **malloc() 함수**를 사용함
 - **malloc.h 파일**을 포함시켜야 한다 (전처리문)
 3. 사용한 후에는 메모리를 해제시켜줌
 - **free(포인터변수) 함수** 사용

malloc() 사용하기

- malloc함수는 메모리를 강제로 설정하므로 프로그램 메모리 해제를 하기 전까지 계속해서 공간을 점유하고 있음
- 따라서, 프로그램이 끝날 때 반드시 free 함수로 메모리를 해제시켜줘야 함

■ 사용방법

포인터 변수 = (포인터 변수의 데이터형*) malloc(포인터 변수의 데이터형 크기 × 필요한 크기)

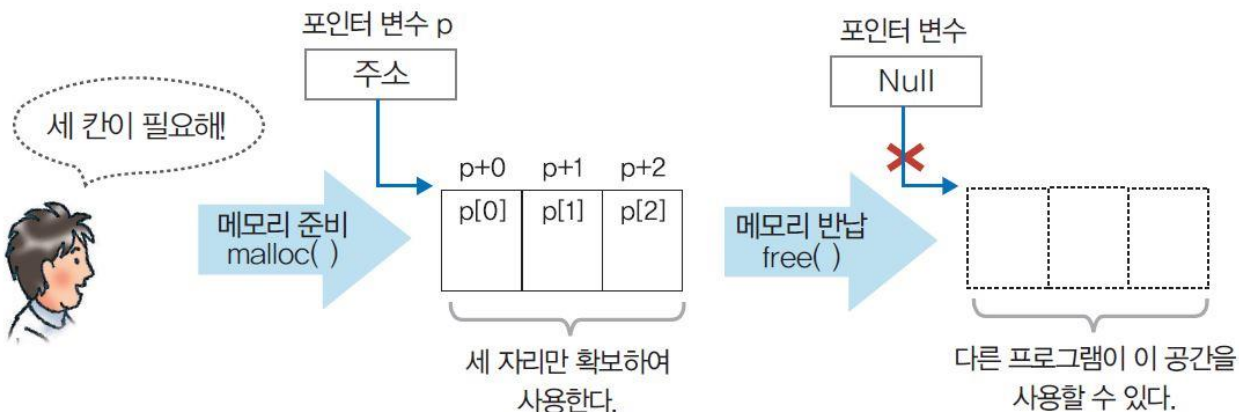


그림 12-3 동적 메모리 할당의 개념

malloc 사용 예시

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

← malloc() 을 포함하는 파일 추가

```
int main() {
```

```
    char *p;
    char oldch, newch;
    int sizeStr;
```

```
    puts("입력할 글자크기를 쓰시오.");
    scanf("%d", &sizeStr);
```

← 할당할 메모리 크기를 사용자에게 물어봄

```
    p = (char*)malloc(sizeof(char) * sizeStr + 1);
```

← malloc() 으로 메모리 크기를 지정
문자열은 끝에 NULL 이 항상
들어가므로 +1 로 크기 설정

```
    puts("여러글자를 입력하시오: ");
    scanf("%s", p);
```

```
    puts("바꿀 기존문자와 새로운 문자를 입력하시오.");
    scanf(" %c, %c", &oldch, &newch); //###scanf 앞에 공백 둘 것!!
```

```
    int i = 0;
    for (i = 0; i < sizeStr; i++) {
        if (p[i] == oldch)
            p[i] = newch;
    }
```

```
    printf("\n 바뀐 글자: %s\n", p);
    free(p);
```

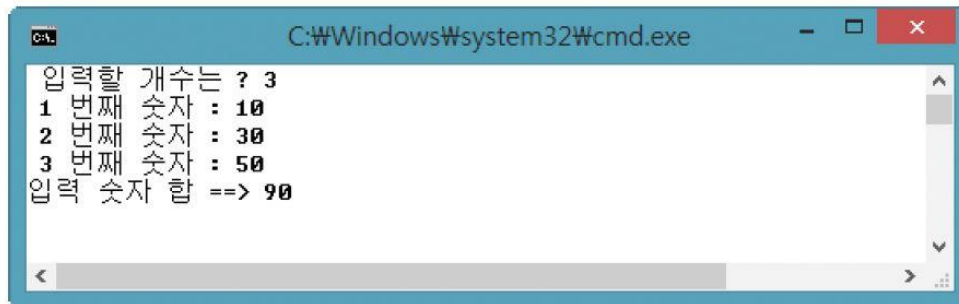
← 할당한 메모리를 해제함

```
}
```

[실습] malloc 을 써봅시다

- 정해지지 않은 개수의 정수를 입력받고 합계 출력하기
 - 정수형 포인터 변수 p를 선언한다.
 - 입력할 숫자의 개수를 물어보고, 변수 cnt에 입력받는다.
 - 위에서 선언한 p에 입력한 개수(cnt)만큼 메모리를 확보한다.
 - 입력한 개수 cnt 만큼 반복하면서 숫자를 입력받는다.
 - 입력받은 숫자의 합계를 누적해서 출력한다.
 - 메모리를 해제한다.

실행 결과 ▼



```
C:\Windows\system32\cmd.exe
입력할 개수는 ? 3
1 번째 숫자 : 10
2 번째 숫자 : 30
3 번째 숫자 : 50
입력 숫자 합 ==> 90
```

다른 메모리 할당 함수

■ calloc()

- 처음부터 0으로 초기화된 메모리를 확보
- 매개변수 두 개로 정의함

포인터변수 = (데이터형*) calloc (데이터형크기, 필요한크기)

■ realloc()

- 처음 확보했던 메모리의 크기를 변경하는 함수
- 사용법: 매개변수 두 개
 - 첫번째 매개변수로 기존포인터 변수가 추가되고 두번째 매개변수는 malloc 함수와 동일함

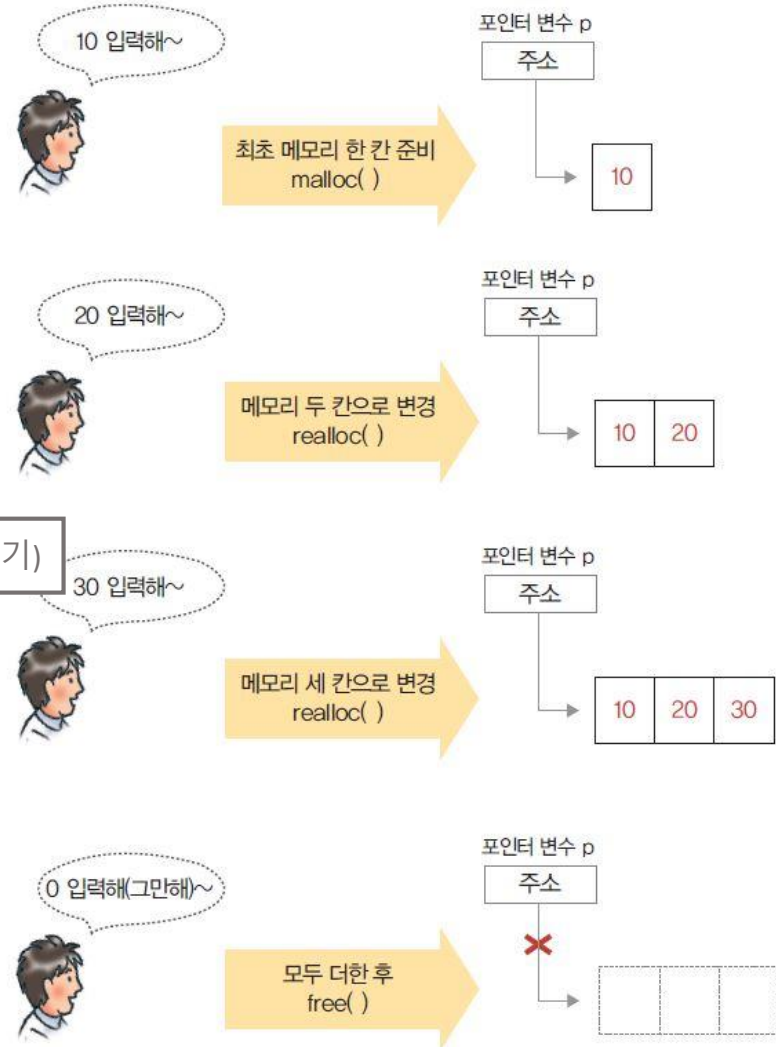


그림 12-4 realloc() 함수의 개념

포인터 변수 = (포인터 변수의 데이터형*) realloc(기본 포인터, 포인터 변수의 데이터형 크기 × 필요한 크기);

p = (int*) realloc(p, sizeof(int) * 10);

realloc 사용 예시

```
int main() {  
  
    int *p;  
    int i = 0, cnt = 0, sum = 0;  
  
    printf("입력할 숫자의 개수: ");  
    scanf("%d", &cnt);  
  
    p = (int*)calloc(sizeof(int), cnt);  
  
    for (i = 0; i < cnt; i++)  
        printf("%d. %d \n", i+1, p[i]);  
  
    printf("다시 입력할 숫자의 개수: ");  
    scanf("%d", &cnt);  
    p = (int*)realloc(p, sizeof(int)*cnt);  
  
    for (i = 0; i < cnt; i++)  
        printf("%d. %d \n", i + 1, p[i]);  
  
    //다시 입력할 숫자만큼 숫자입력받고 합계출력하기  
  
    free(p);  
}
```

← calloc() 으로 메모리 크기를
지정하면서 초기값이 0으로 설정

← realloc() 으로 메모리 크기 조정

← 메모리 해제

Today

1. 지난 시간 복습 – 전처리문과 구조체
2. 메모리 할당하기
3. 파일입출력
4. 실습

표준 입출력

■ 표준 입출력

- 키보드로 입력
- 화면(모니터로) 출력

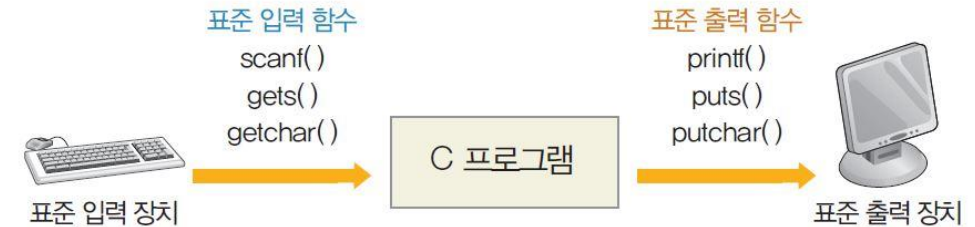


그림 11-1 표준 입출력의 개념

■ 서식화된 입출력

- printf(), scanf(): 모든 형식의 데이터 입출력 가능

구문	설명
printf("서식", 출력할 매개변수들 ...)	표준 출력 장치(모니터)에 서식을 맞춰 출력한다.
scanf("서식", 입력할 매개변수들 ...)	표준 입력 장치(키보드)에서 서식에 맞춰 입력받는다.

■ 문자열 입출력

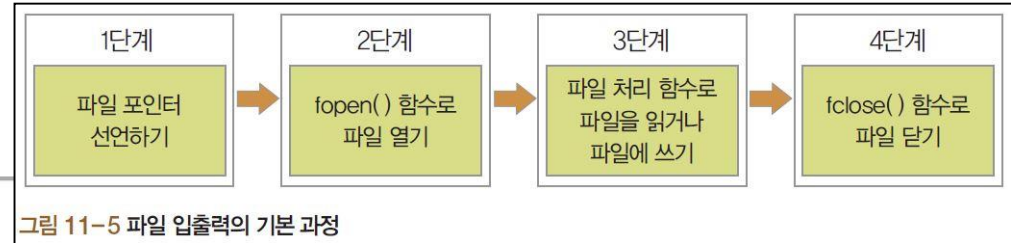
- puts(), gets(): 문자열만 입출력

파일 입출력

■ 파일 입출력 기본 과정

- 1단계: 파일 포인터 선언

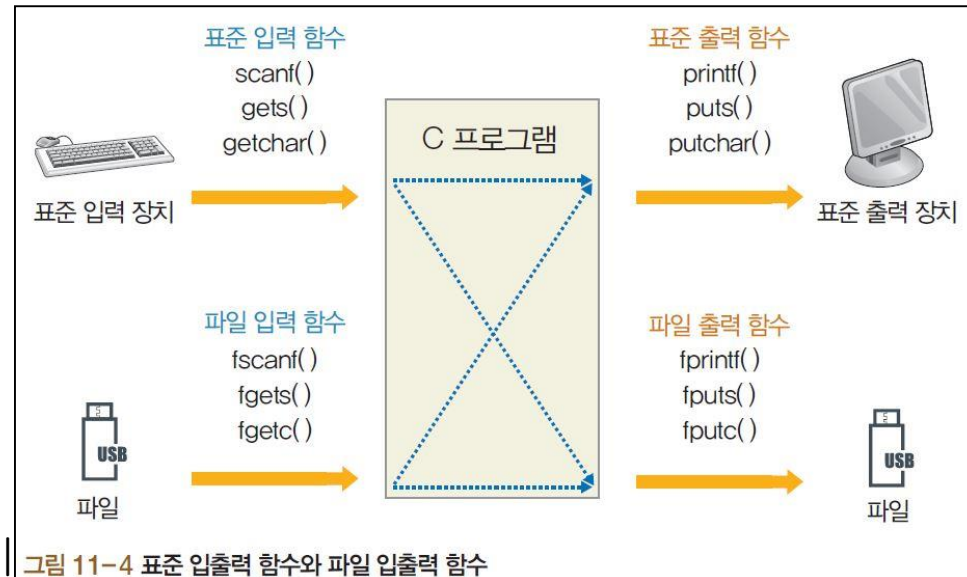
```
FILE *변수 이름;
```



- 2단계: fopen 함수로 파일 열기

```
변수 이름 = fopen("파일 이름", "열기 모드");
```

- 3단계 : 파일 처리 함수로 파일을 읽거나 파일에 쓰기



- 4단계 : fclose() 함수로 파일 닫기

```
fclose(파일 포인터);
```


Today

1. 지난 시간 복습 – 전처리문과 구조체
2. 메모리 할당하기
3. 파일입출력
4. 실습

[실습2]

- ▶ **int**형 배열에 0에서 999까지 **random number**를 100개 생성하여 저장한 후, 100단위별로 생성된 숫자들의 빈도수를 출력해 보세요.
- ▶ **Hint.** 앞의 예제를 확장하여 빈도수를 저장할 크기가 10인 **int**형 배열 **bb**를 생성하여 모두 0으로 초기화한다. 다음으로 **for**문을 사용하여 100개의 숫자들을 하나씩 반복하면서 각각 10개의 구간에 속하는 지를 체크하여, 속하는 구간의 빈도수를 1씩 증가시킨다.

0	-	99	:	10개
100	-	199	:	14개
200	-	299	:	8개
300	-	399	:	7개
400	-	499	:	9개
500	-	599	:	12개
600	-	699	:	9개
700	-	799	:	8개
800	-	899	:	12개
900	-	999	:	11개

[실행 예]

[실습] 배열 내 모든 원소의 합 구하기

- 개수를 나타내는 변수 N 을 전처리기로 정의하고 초기값은 10으로 설정하기
- int 형 배열 N개 원소를 담도록 생성

=====다음의 1~4는 모두 같은 결과를 나타냄=====

1. 반복문 안에서 배열을 사용해서 원소 N개 전체 합계를 계산하고 출력
2. 반복문 안에서 포인터를 사용해서 원소 전체 합계를 계산하고 출력
3. 합계계산하는 부분을 함수로 만들어서 계산하고 출력하기(함수명: sumTotal, 함수반환값: 정수형, 매개변수:없음)
4. 합계계산하는 부분을 함수로 만들어서 계산하고 출력하기(함수명: sumTotal, 함수반환값: 정수형, 매개변수:없음)