7장 배열

김명호

내용

- 1차원 배열
- 배열의 초기화
- 다차원 배열
- 배열과 함수
- 문자열
- 문자열처리 라이브러리 함수
- 가변길이 배열

배열

- 성적 3개를 정렬하는 코드

```
int grade0, grade1, grade2;
if (grade0 <= grade1)
   if (grade1 <= grade2)
      printf("%d, %d, %d\n", grade0, grade1, grade2);
   else if (grade0 <= grade2)
      printf("%d, %d, %d\n", grade0, grade2, grade1);
   else
      printf("%d, %d, %d\n", grade2, grade0, grade1);
else
   if (grade0 <= grade2)
      printf("%d, %d, %d\n", grade1, grade0, grade2);
   else if (grade1 <= grade2)
      printf("%d, %d, %d\n", grade1, grade2, grade0);
   else
      printf("%d, %d, %d\n", grade2, grade1, grade0);
```

배열

- 대량의 데이터를 다루어야 할 때 일반 변수 만을 사용하면 프로그래밍이 어려워 짐
- 배열은 같은 형의 많은 데이터를 다루어야 할 때 유용
- 배열의 원소는 인덱스로 지정됨

1차원 배열

- 선언

  형 배열이름[배열크기];

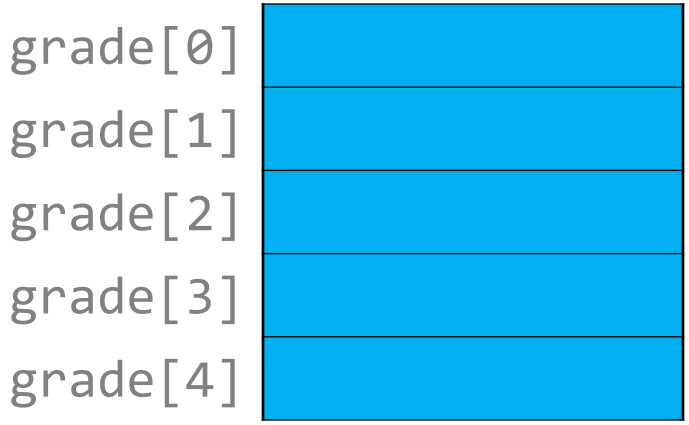
  - 형 : 데이터 자료형
  - 배열이름 : 식별자
  - 배열크기 : 원소 개수
  - 배열 원소 인덱스 : 0부터 배열크기 -1
  - 인덱스는 정수적 수식을 사용해야 함

1차원 배열

- **선언 예제**

```
int  grade[5];
```

  - 메모리에 int 형 변수 5개 할당



1차원 배열

- **사용 예제**

```
int  grade[5], x, i, sum;
float average;

grade[2] = 3;        // grade[2]에 3을 배정함
x = grade[i+2];      // grade[i]의 값을 x에 배정함

// 배열은 반복문 내에서 주로 사용됨
for (i = sum = 0; i < 5; i++)
   sum += grade[i];
average = sum / 5.0;
```

1차원 배열

프로그램 7.1 일부

```
int grade[10], i, j, sum, tmp;
for (i = 0; i < 10; i++)
   scanf("%d", &grade[i]);
// 버블 정렬
for (i = 0; i < 9; i++)
   for (j = 9; j > i; j--)
      if (grade[j - 1] > grade[j]){
         tmp = grade[j - 1];
         grade[j - 1] = grade[j];
         grade[j] = tmp;
      }
for (i = 0; i < 10; i++)
   printf("%d ", grade[i]);
```

1차원 배열

- **버블 정렬 (내부 for 문)**

	j=9	j=8	j=7	j=6	j=5	j=4	j=3	j=2	j=1
grade[0]	80	80	80	80	80	80	80	80	60
grade[1]	90	90	90	90	90	90	90	60	80
grade[2]	99	99	99	99	99	99	60	90	90
grade[3]	77	77	77	77	77	60	99	99	99
grade[4]	89	89	89	89	60	77	77	77	77
grade[5]	60	60	60	60	89	89	89	89	89
grade[6]	88	88	83	83	83	83	83	83	83
grade[7]	83	83	88	88	88	88	88	88	88
grade[8]	94	94	94	94	94	94	94	94	94
grade[9]	97	97	97	97	97	97	97	97	97

1차원 배열

- **버블 정렬 (외부 for 문)**

		i=0	i=1	i=2	i=3	i=4	i=5	i=6	i=5	i=5
grade[0]	80	60	60	60	60	60	60	60	60	60
grade[1]	90	80	77	77	77	77	77	77	77	77
grade[2]	99	90	80	80	80	80	80	80	80	80
grade[3]	77	99	90	83	83	83	83	83	83	83
grade[4]	89	77	99	90	88	88	88	88	88	88
grade[5]	60	89	83	99	90	89	89	89	89	89
grade[6]	88	83	89	88	99	90	90	90	90	90
grade[7]	83	88	88	89	89	99	94	94	94	94
grade[8]	94	94	94	94	94	94	99	97	97	97
grade[9]	97	97	97	97	97	97	97	99	99	99

기호 상수

- 10개의 성적을 다루는 앞 프로그램을 30개 성적을 다루는 것으로 수정해야 한다면, 프로그램에서 모든 10은 30으로 9는 29로 수정하면 됨
  - 대형 프로그램 에서는 매우 어려운 작업 임
- 배열을 사용할 기호 상수를 사용하면 유용함
- 기호 상수는 #define을 사용하여 선언 함
- 예

```
#define   N   10
```

  - N : 기호 상수
  - N은 프로그램에서 상수 처럼 사용됨

기호 상수

프로그램 7.2 일부

```
#define  N   10    // 데이터 크기를 변경할 때 여기만 수정하면 됨
int grade[N], i, j, sum, tmp;

for (i = 0; i < N; i++)
   scanf("%d", &grade[i]);

for (i = 0; i < N - 1; i++)
   for (j = N - 1; j > i; j--)
      if (grade[j - 1] > grade[j]){
         tmp = grade[j - 1];
         grade[j - 1] = grade[j];
         grade[j] = tmp;
      }
```

배열의 초기화

- 선언문에서 중괄호 내에 값들을 나열 하여 초기화함
- 초기화 예제

```
int grade[5] = {100, 90, 77, 100, 50};
```

  - grade[0] = 100, grade[1] = 90, . . .

- 초기자 목록이 초기화되는 배열 원소 개수보다 적다면, 나머지 원소들은 0으로 초기화됨

```
int grade[5] = {100, 90, 77};
```

  - grade[3] = 0, grade[4] = 0

배열의 초기화

- 초기화자가 명시된 배열 선언에서 배열 크기는 생략될 수 있음
  - 초기자의 개수가 배열의 암시적인 크기가 됨

```
int grade[] = {100, 90, 77, 100, 50};
```

    - 배열 크기로 5가 생략된 것과 같음

배열의 초기화

- **특정 원소 초기화**
  - C99
  - 초기화 목록에 각괄호로 명시함

```
int  grade[5] = { [2] = 80, [4] = 90 };
// grade[2] = 80, grade[4] = 90
```

배열의 초기화

- 일반 수식 초기화자
  - C99
  - C90에서 초기화자는 상수 수식만 올 수 있지만, C99에서는 일반 수식이 올 수 있음

```
int value = 50;

int grade[] = {value, value * 2, value + 30};
```

배열의 초기화

프로그램 7.3 일부

```
#define N 10
int main(void) {
   int fib[N] = {0, 1}, i;
   // 피보나치수열 생성
   for (i = 2; i < N; i++)
      fib[i] = fib[i - 1] + fib[i - 2];
   // 피보나치 출력
   printf("피보나치 수 : ");
   for (i = 0; i < N; i++)
      printf("%d ", fib[i]);
   printf("\n");
   return 0;
}
```

다차원 배열

- 임의의 형의 배열이 가능함
- 다차원 배열
  - 2차원 배열 : 1차원 배열의 1차원 배열
  - 3차원 배열 : 2차원 배열의 1차원 배열
- 선언문에서 각괄호의 개수에 의해 차원이 결정됨

배열 선언	차원
int a[100];	1차원 배열
int b[2][7];	2차원 배열
int c[5][3][2];	3차원 배열

2차원 배열

- 2차원 배열은 행과 열을 갖는 직사각형의 원소의 집합으로 생각하는 것이 편리함
- 많은 데이터들이 2차원으로 표현됨

과목명 / 이름	화학	수학	물리	국어
학생 0	98	70	95	85
학생 1	88	80	83	90
...	...	...	...	...
학생 49	77	60	90	86

2차원 배열

- **선언**

```
int b[2][7];
// int [7]을 원소로 2개 갖는 1차원 배열
```

b	[0]	[1]	[2]	[3]	[4]	[5]	[6]
b[0]							
b[1]							

- **각 원소의 인덱스**

b	[0]	[1]	[2]	[3]	[4]	[5]	[6]
b[0]	b[0][0]	b[0][1]	b[0][2]	b[0][3]	b[0][4]	b[0][5]	b[0][6]
b[1]	b[1][0]	b[1][1]	b[1][2]	b[1][3]	b[1][4]	b[1][5]	b[1][6]

2차원 배열

프로그램 7.4 일부

```
#define N 3   // 학생 수
#define M 4   // 과목 수
int main(void){
   int grades[N][M], student[N] = {0}, i, j;
   for (i = 0; i < N; ++i)
      for (j = 0; j < M; ++j)
         scanf("%d", &grades[i][j]);
   for (i = 0; i < N; ++i){
      for (j = 0; j < M; ++j)
         student[i] += grades[i][j];
      printf("%1d 평균 : %5.2f\n", i, (float)student[i] / M);
   }
}
```

3차원 배열

프로그램 7.5 일부

```
#define N 3      // 학생 수
#define M 4      // 과목 수
#define K 2      // 학기 수
. . .
int grades_semesters[N][K][M], i, j, k;
float student[N][K] = {0.0};
for (i = 0; i < N; ++i)
   for (j = 0; j < K; ++j){
      for (k = 0; k < M; ++k)
         student[i][j] += grades_semesters[i][j][k];
      student[i][j] /= M;
      printf("%1d %1d학기 평균: %5.2f\n", i, j, student[i][j]);
   }
```

다차원 배열 초기화

- 초기화에 중괄호 사용하면 편리함

```
int   b[2][3] = {1, 2, 3, 4, 5, 6};
int   b[2][3] = {{1, 2, 3}, {4, 5, 6}};
int   b[ ][3] = {{1, 2, 3}, {4, 5, 6}};
```

  - 내부 중괄호가 없으면, 배열은 b[0][0], b[0][1], ..., b[1][2] 순으로 초기화되고, 인덱싱은 행 우선 임
  - 배열의 원소 수보다 더 적은 수의 초기화 값이 있다면, 남는 원소는 0으로 초기화됨
  - 첫 번째 각괄호가 공백이면, 컴파일러는 내부 중괄호 쌍의 수를 그것의 크기로 함
  - 첫 번째 크기를 제외한 모든 크기는 명시해야 함

다차원 배열 초기화

- 3차원 배열 초기화 예

```
int  c[2][2][3]={
        {{1, 1, 0}, {2, 0, 0}},
        {{3, 0, 0}, {4, 4, 0}}
     };
```

- 초기화자의 개수가 적으면 나머지 원소는 0으로 초기화 됨

```
int  c[][2][3]={{{1, 1}, {2}}, {{3}, {4, 4}}};
```

다차원 배열 초기화

- 모든 배열 원소를 0으로 초기화 하기

```
int  b[2][3] = {0};
int  c[2][2][3] = {0};
```

다차원 배열 초기화

- 특정 원소 초기화
  - C99
  - 초기화 목록에 인덱스로 명시함

```
int d[3][2][3] = { [2][1][0] = 9,
                   [2][1][1] = 10 };
// d[2][1][0] = 9, d[2][1][1] = 10
```

배열과 함수

- **1차원 배열과 함수**
  - 배열에 저장된 N 개의 성적을 더하는 함수

함수 7.1

```
int grade_sum(int gr[N]){
    int sum, i;
    for (sum = 0, i = 0; i < N; i++)
        sum += gr[i];
    return sum;
}
```

배열과 함수

- grade_sum() 함수 호출

  sum = grade_sum(grade[N]);

  – grade_sum() 매개변수와 인자의 형이 일치하지 않음

    - 매개변수(int gr[N]) : int 형 배열
    - 인자(grade[N]) : int 형

  sum = grade_sum(grade);

  – 배열 이름만 명시하면 됨

    - 포인터를 배우면서 이것에 대한 자세한 설명이 있을 것임

배열과 함수

- 임의의 크기 배열을 다룰 수 있게 함수를 작성하는 것이 좋음
  - 배열 크기를 인자로 명시하면 됨

함수 7.2

```
int grade_sum2(int gr[], int size){
    int sum, i;
    for (sum = 0, i = 0; i < size; i++)
        sum += gr[i];
    return sum;
}
```

배열과 함수

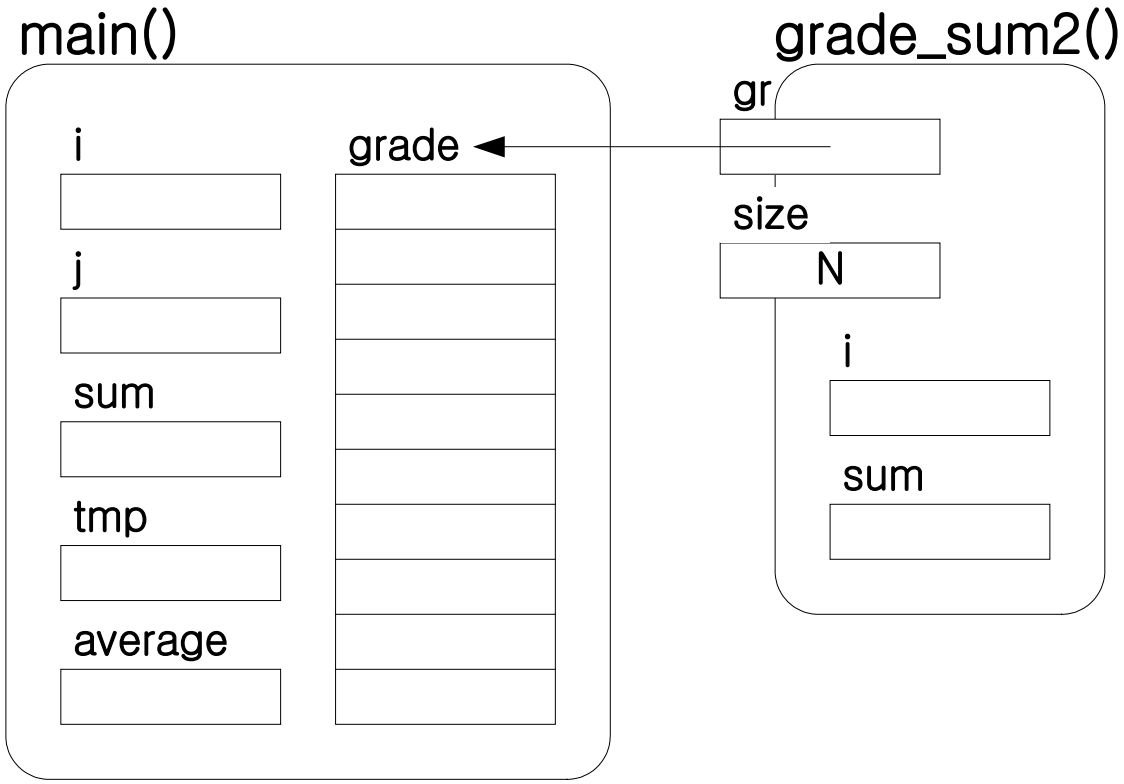
- grade_sum2() 함수 호출

```
sum = grade_sum(grade, N);
```

  - 배열 이름과 크기를 인자로 하여 호출

sum = grade_sum2(grade, N);

- grade_sum2()의 gr은 사실 main()의 grade 배열을 포인트 하고 있음 (gr == grade)



배열과 함수

- 함수 헤더에서 1차원 배열 매개변수의 배열 크기는 생략해도 됨
- 다음 두 헤더는 같은 의미임

```
int grade_sum2(int gr[N], int size)
int grade_sum2(int gr[], int size)
```

배열과 함수

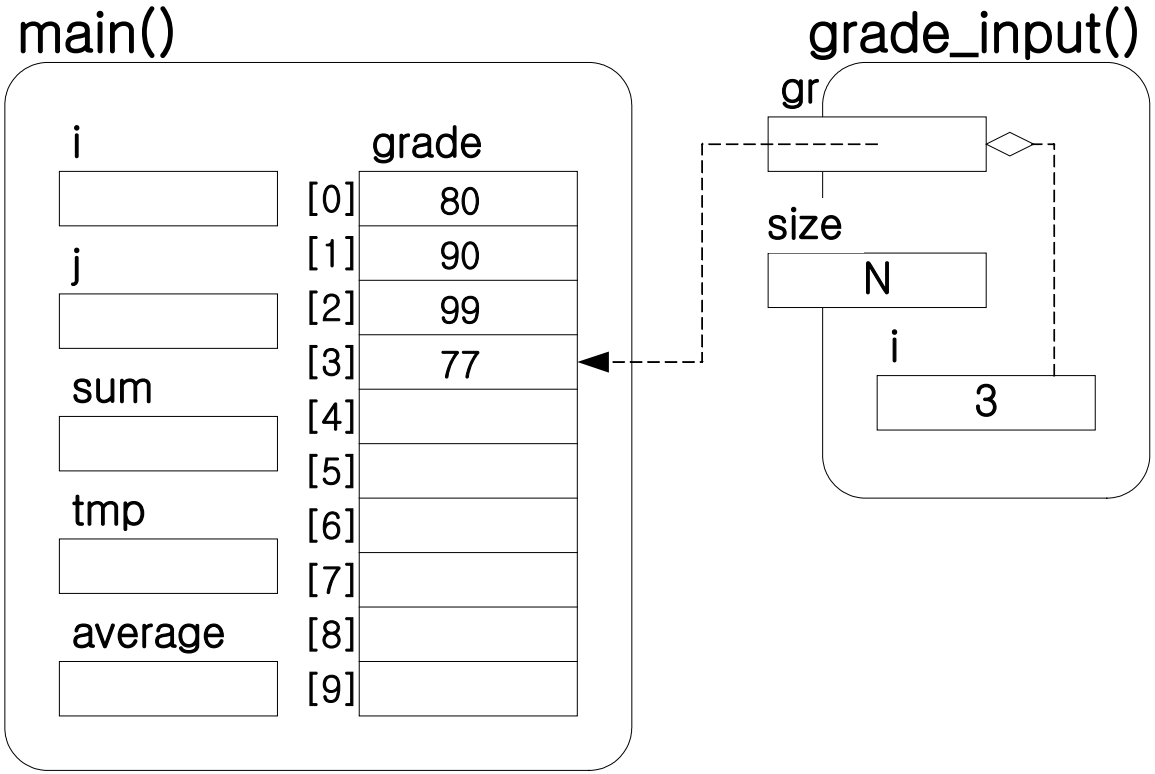
- **성적 입력 함수**

함수 7.3

```
void grade_input(int gr[], int size){
    int i;
    for (i = 0; i < size; i++) {
        printf("%d 번째 성적을 입력하세요 : ", i);
        scanf("%d", &gr[i]);
    }
}
```

grade_input(grade, N);

- gr[i]에 값을 입력하는 것은 사실 grade[i]에 입력하는 것임



배열과 함수

프로그램 7.6 일부

```
#define N 10
void grade_input(int gr[], int size);
int grade_sum2(int gr[], int size);
void grade_sort(int gr[], int size);

int main(void){
    int grade[N], i, sum;
    float average;
    grade_input(grade, N);
    sum = grade_sum2(grade, N);
    average = (float) sum / N;
    printf("성적 평균 : %f\n", average);
    grade_sort(grade, N);
    printf("성적순 : ");
    for (i = 0; i < N; i++) printf("%d ", grade[i]);
}
```

다차원 배열과 함수

- 2차원 배열을 다루는 함수

함수 7.4

```
void grades_sum(int gr[N][M], int st[], int size)
{
    int i, j;
    for (i = 0; i < size; ++i){
        st[i] = 0;
        for (j = 0; j < M; ++j)
            st[i] += gr[i][j];
    }
}
```

배열과 함수

- void grades_sum(int [N][M], int [], int) 호출

  프로그램 7.4 성적 평균 구하는 부분 수정

```
int grades[N][M], student[N];
grades_sum(grades, student, N);
for (i = 0; i < N; i++)
    printf("학생 %1d 평균 : %5.2f\n", i,
                        (float)student[i] / M);
```

배열과 함수

- 함수 헤더에서 배열 매개변수의 첫 번째 배열 크기는 생략해도 됨
- 다음 두 헤더는 같은 의미 임

```
void grades_sum(int gr[N][M], int st[], int size)
void grades_sum(int gr[][M], int st[], int size)
```

성적 배열

grades[N][M]

이름 \ 과목명	화학	수학	물리	국어
학생 0	98	70	95	85
학생 1	88	80	83	90
...	...	...	...	...
학생 49	77	60	90	86

student[N]

총 합
0
0
...
0

grades_sum(grades, student, N);

함수 7.4

```
void grades_sum(int gr[N][M], int st[], int size)
{
    int i, j;
    for (i = 0; i < size; ++i){
        st[i] = 0;
        for (j = 0; j < M; ++j)
            st[i] += gr[i][j];
    }
}
```

grades_sum(grades, student, N);

grades[N][M]

이름 \ 과목명	화학	수학	물리	국어
학생 0	98	70	95	85
학생 1	88	80	83	90
...	...	...	...	...
학생 49	77	60	90	86

student[N]

총 합
348
341
...
313

다차원 배열과 함수

- 2차원 배열을 다루는 함수

함수 7.5

```
int grade_sum_id(int gr[][M], int id){
    int i, sum = 0;
    for (i = 0; i < M; i++)
        sum += gr[id][i];
    return sum;
}
```

```
student[id] = grade_sum_id(grades, id);  // id == 1
```

grades[N][M]

이름 \ 과목명	화학	수학	물리	국어
학생 0	98	70	95	85
학생 1	88	80	83	90
...	...	...	...	...
학생 49	77	60	90	86

student[N]

총 합
348
341
...

배열과 함수

- void grade_sum_id(int [][M], int id) 호출

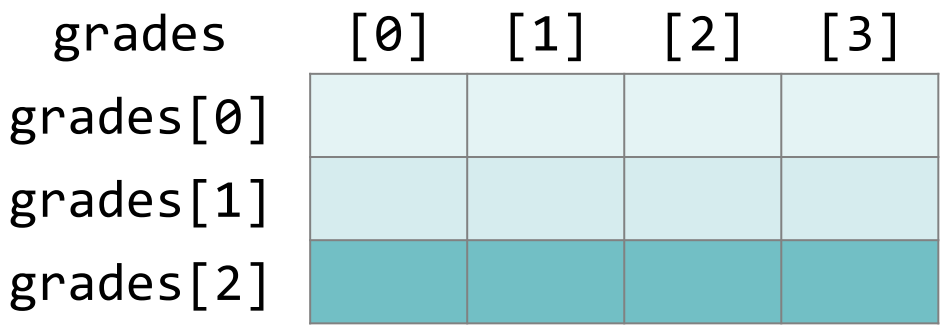
프로그램 7.4 성적 평균 구하는 부분 수정

```
int grades[N][M], student[N];
for (i = 0; i < N; ++i){
    student[i] = grade_sum_id(grades, i);
    printf("학생 %1d 평균 : %5.2f\n", i,
                                (float)student[i] / M);
}
```

배열과 함수

- 2차원 배열인 grades가 grade_sum_id()로 전달되지만 호출되면 i 행만 사용됨

```
for (i = 0; i < N; ++i)
    student[i] = grade_sum_id(grades, i);
```



→ 1차원 배열을 더하는 grade_sum2() 함수를 사용해도 됨

grade_sum2()

함수 7.2

```
int grade_sum2(int gr[], int size){
    int sum, i;
    for (sum = 0, i = 0; i < size; i++)
        sum += gr[i];
    return sum;
}
```

배열과 함수

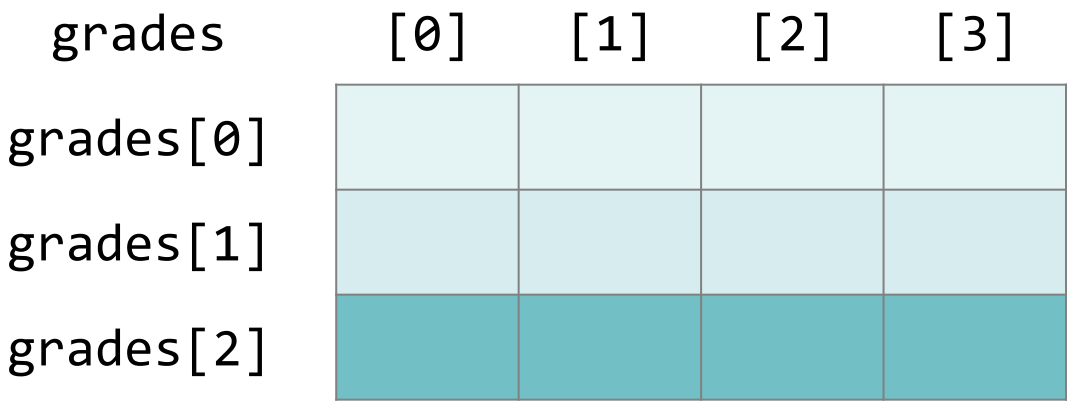
- void grade_sum2(int [], int) 호출

프로그램 7.4 성적 평균 구하는 부분 수정

```
int grades[N][M], student[N];
for (i = 0; i < N; ++i){
    student[i] = grade_sum2(grades[i], M);
    printf("학생 %1d 평균 : %5.2f\n", i,
                                (float)student[i] / M);
}
// grades[i] : grades의 i번째 행
```

배열과 함수

- **int grades[N][M];**
  - grades[0][0], grades[0][1],
  - grades[0]
  - grades



문자열

- **문자열**
  - char 형의 1차원 배열
  - 문자열은 끝의 기호인 \0, 또는 널 문자로 끝남
  - 널 문자 : 모든 비트가 0인 바이트; 십진 값 0
  - 문자열의 크기는 \0까지 포함한 크기

문자열

- **문자열 상수**
  - 큰따옴표 안에 기술됨
  - 문자열 예 : "good"
    - 마지막 원소가 널 문자이고 크기가 5인 문자 배열
- **주의 - "a"와 'a'는 다름**
  - 배열 "a"는 두 원소를 가짐
  - 첫 번째 원소는 'a', 두 번째 원소는 '\0'

문자열

- char 형 1차원 배열 초기화
  - 일반 배열과 같은 방법으로 초기화

```
char s[8] = {'p', 'r', 'o', 'g', 'r', 'a', 'm', '\0'};
```

  - 문자열 상수로 초기화

```
char s[8] = "program";
```

문자열

- char 형 1차원 배열 초기화
  - 초기화자가 있으면 배열 크기를 생략할 수 있음

```
char t[] = "C program";     // 배열 크기 : 10
```

t[0]	'C'
t[1]	' '
t[2]	'p'
t[3]	'r'
t[4]	'o'
t[5]	'g'
t[6]	'r'
t[7]	'a'
t[8]	'm'
t[9]	'\0'

문자열

- 문자열은 char 형 1차원 배열이기 때문에 문자열 상수 뒤에 첨자를 붙일 수 있음
  - "program"[0] : 'p'
  - "program"[4] : 'r'

문자열 출력

프로그램 7.7

```
#include <stdio.h>
#define N 10
int main(void){
    char str[N] = "Hello!";
    int i;
    for (i = 0; i < N; i++)
        if (str[i] == '\0')
            break;
        else
            printf("%c", str[i]);
    printf("\n");
    return 0;
}
```

문자열 출력

프로그램 7.8

```
#include <stdio.h>
#define N 10
int main(void){
    char str[N] = "Hello!";
    printf("%s\n", str);
    return 0;
}
```

문자열 입력

프로그램 7.9

```
#include <stdio.h>
#define N 20
int main(void){
    char name[N] = "";
    int age = 100;
    printf("이름을 입력하세요 : ");
    scanf("%s", name);
    printf("나이를 입력하세요 : ");
    scanf("%d", &age);
    printf("%s는(은) %d 살입니다.\n", name, age);
    return 0;
}
```

문자열 입력

- 프로그램 7.9에서 이름을 입력 받을 때 문제점
  - 이름은 19자 이하여야 함
  - %s로 입력 받으면 공백문자 직전까지만 입력됨

프로그램 결과

```
$ name
이름을 입력하세요 : 김진혁
나이를 입력하세요 : 8
김진혁는(은) 8 살입니다.
$ name
이름을 입력하세요 : 김 진혁
나이를 입력하세요 : 김는(은) 100 살입니다.
```

문자열 입력

- 입력 문자열의 길이는 %와 s사이에 명시함

```
char str1[5], str2[8];
scanf("%4s%7s", str1, str2);
```

  - str1에는 최대 4개의 문자가 입력되어 저장되고
  - str2에는 최대 7개의 문자가 입력되어 저장됨
  - "GoodMorningWorld"이 입력된 경우
    - str1 : "Good"
    - str2 : "Morning"

문자열 입력

- 입력 오류 처리를 위해서는 getchar()를 사용하는 것이 좋음
  - <stdio.h>에 정의되어 있고
  - 입력 스트림에서 문자를 하나 읽음

문자열 입력

프로그램 7.10 이름 입력 부분

```
printf("이름을 입력하세요 : ");
i = 0;
while (1) {
   c = getchar();
   if ((c == ' ') || (i == N)) {    // 공백이나 N보다 크면 다시 입력받음(오류처리)
      while ((c = getchar()) != '\n')   ;  // 개행문자까지 무시 함
      printf("이름을 다시 입력하세요 : ");
      i = 0;
      continue;
   }
   if ((c == '\n') || (c == EOF)) {        // 이름 읽기 종료
      name[i++] = '\0';
      break;
   }
   name[i++] = c;          // 이름 문자를 name에 저장
}
```

프로그램 결과

```
$ name
이름을 입력하세요 : 김 진혁
이름을 다시 입력하세요 : 김진혁
나이를 입력하세요 : 20
김진혁는(은) 20 살입니다.
```

문자열 입력

프로그램 7.11 input_name() 함수

```
#define N 21
void input_name(char name[]){
   int i = 0, c;
   printf("이름을 입력하세요 : ");
   while (1) {
      c = getchar();
      if ((c == ' ') || (i == N)) {
         while ((c = getchar()) != '\n')   ;
         printf("이름을 다시 입력하세요 : ");
         i = 0;
         continue;
      }
      if ((c == '\n') || (c == EOF)) {
         name[i++] = '\0';
         break;
      }
      name[i++] = c;
   }
}
```

문자열 입력

프로그램 7.11

```
#include <stdio.h>
#define N 21
void input_name(char []);
int main(void){
    char name[N] = "";
    int age = 100;
    // 이름을 입력 받음
    input_name(name);
    printf("나이를 입력하세요 : ");
    scanf("%d", &age);
    printf("%s는(은) %d 살입니다.\n", name, age);
    return 0;
}
```

문자열처리 라이브러리 함수

- <string.h>에 문자열를 다루는 다양한 함수가 정의되어 있음
- 많이 사용하는 함수
  - 문자열 복사 : strcpy()
  - 문자열 길이 : strlen()
  - 문자열 비교 : strcmp()
  - 문자열 연결 : strcat()

문자열 복사

```
#include <stdio.h>
#define N 10
int main(void){
    char str[N] = "Hello!";
    char str2[N];
    str2 = str;        // ???
    str2 = "Bond";     // ???
    return 0;
}
```

문자열 복사

- char *strcpy(char * restrict DST,
  const char * restrict SRC);
  - SRC의 문자열을 \0이 나올 때까지 DST에 복사
  - DST의 크기는 SRC를 복사할 만큼 충분히 커야 함
  - 예

```
strcpy(name, "Bond");
strcpy(name, name2);
```

문자열 복사

- **sprintf()를 통해서도 할 수 있음**

```
int sprintf(char *s, const char *ctr_str, ...);
```

  - 출력 결과가 s에 들어가는 것을 제외하면 printf()와 같음
  - 예

```
sprintf(name, "Bond");
sprintf(name, "%s", name2);
```

문자열 길이

- size_t strlen(const char *STR);
  - \0을 뺀 문자의 개수를 리턴
  - 예

```
char name[10] = "James Kim", *name2;
int size = 0;
size = strlen(name);

if (sizeof(name) >= strlen(name2) + 1)
    strcpy(name, name2);
else
    printf("문자열 복사 오류입니다.\n");
```

문자열 비교

- 두 문자열을 등가나 관계 연산자로 비교할 수 없음

```
if (str1 == str2)
    printf("두 문자열은 같습니다.\n");
```

  - 이것은 str1과 str2의 주소를 비교하는 것임
  - str1과 str2가 포인트하는 문자열을 비교하기 위해서는 다른 방법을 사용해야 함

문자열 비교

- **반복문을 통한 두 문자열을 비교**

```
for (i = 0; i < N; i++)
    if (str1[i] == str2[i]){          // 한 문자씩 비교
        if (str1[i] == '\0'){
            printf("두 문자열은 같습니다.\n");
            break;
        }
    }
    else{
        printf("두 문자열은 같지 않습니다.\n");
        break;
    }
```

문자열 비교

- int strcmp(const char *s1, const char *s2);
  - s1과 s2를 사전적 순서로 비교하여, s1이 작으면 음수, 크면 양수, 같으면 0을 리턴
  - 예

```
char str1[N] = "pear", str2[N] = "apple";
int i = 0;
i = strcmp(str1, str2);  // i에 양수 저장됨
```

문자열 연결

- char *strcat(char * restrict DST,
  const char * restrict SRC);
  - DST 문자열 뒤에 SRC 문자열을 결합하여 DST에 저장
  - 예

```
char str1[N] = "Apple", str2[N] = "s";
strcat(str1, str2);
printf("%s\n", str1);      // Apples 출력
```

문자열을 숫자로 변환

- <stdlib.h>에 다양한 함수 선언
  - ato...()와 strto...() 부류 함수
  - ato...() : 문자열을 10진수로 변환

```
value = atoi("123.456");
// value에 123 저장
```

  - strto...() : 변환 진수와 길이 지정 가능

문자열을 숫자로 변환

- ato...() 부류 함수

함수	변환 형
atof()	double
atoi()	int
atol()	long
atoll()	long long

ato...()

```
#include <stdio.h>
#include <stdlib.h>
int main(){
   int  i_val;
   float f_val;
   char str1[10] = “123.456", str2[10] = "Cygwin";

   i_val = atoi(str1);
   f_val = atof(str1);
   printf("String : %s, int = %d, float = %f\n", str1, i_val, f_val);
   i_val = atoi(str2);
   f_val = atof(str2);
   printf("String : %s, int = %d, float = %f\n", str2, i_val, f_val);
   return(0);
}
```

ato...()

```
$ ./a
String : 123.456, int = 123, float = 123.456000
String : Cygwin, int = 0, float = 0.000000
```

문자열을 숫자로 변환

- strto...() 부류 함수

함수	변환 형
strtod()	double
strtof()	float
strtold()	long double
strtol()	long
strtoll()	long long
strtoul()	unsigned long
strtoull()	unsigned long long

Strto…()

```
#include <stdio.h>
#include <stdlib.h>
int main(){
   char str[30] = "123456789a value", char **ptr;
   long value;

   ptr = malloc(4);
   value = strtol(str, ptr, 10);
   printf("Number : %ld, Rest part : %s\n", value, *ptr);
   value = strtol(str, ptr, 8);
   printf("Number : %ld, Rest part : %s\n", value, *ptr);
   value = strtol(str, ptr, 16);
   printf("Number : %ld, Rest part : %s\n", value, *ptr);
   return(0);
}
```

Strto…()

```
$ ./a
Number : 123456789, Rest part : a value
Number : 342391, Rest part : 89a value
Number : 2147483647, Rest part :  value
```

가변길이 배열

- 배열 선언에서 배열 크기를 정수 상수 수식이 아니라 정수 수식으로 지정한 배열
- C99에 추가
- 배열 크기에 유연한 프로그램 작성이 가능함
- 동적 메모리 할당 기법을 사용하여 작성할 수도 있음

문자열 입력

프로그램 7.13 일부

```
#include <stdio.h>
int main(void){
    int N;     // 변수 N, 기호 상수 아님
    int i, j, sum, tmp;
    float average;
    printf("처리할 성적 개수를 입력하세요 : ");
    scanf("%d", &N);
    int grade[N]; // 가변길이 배열
    // 성적 입력
    for (i = 0; i < N; i++) {
        printf("%d 번째 성적을 입력하세요 : ", i);
        scanf("%d", &grade[i]);
    }
```

프로그램 결과

```
$ program6_13
처리할 성적 개수를 입력하세요 : 0
0보다 커야 합니다. 다시 입력하세요 : -20
0보다 커야 합니다. 다시 입력하세요 : 5
0 번째 성적을 입력하세요 : 100
1 번째 성적을 입력하세요 : 98
2 번째 성적을 입력하세요 : 78
3 번째 성적을 입력하세요 : 88
4 번째 성적을 입력하세요 : 90
성적 평균 : 90.800003
성적순 : 78 88 90 98 100
$ program6_13
처리할 성적 개수를 입력하세요 : 3
0 번째 성적을 입력하세요 : 89
1 번째 성적을 입력하세요 : 90
2 번째 성적을 입력하세요 : 100
성적 평균 : 93.000000
성적순 : 89 90 100
```