# 프로그래밍 1 Lecture Note #11

백윤철 ybaek@smu.ac.kr

# 내용

- □ 배열의 이해
- □ 배열의 정의
- □ 배열의 초기화
- □ 문자 배열과 문자열

#### 배열이란 무엇인가?

- □ 수업에 있는 모든 학생들의 성적을 입력해서 관리 하는 프로그램을 작성한다고 가정해보자
- □ 중간고사 (30%), 기말고사 (35%), 과제 (30%), 출석 (5%) 등을 관리해야 함
- □ 40명 정도의 학생들이 있음
- □ 따라서 변수를 만들어서 관리한다면 int midterm01, midterm02, ..., midterm40; int final01, final02, ..., final40;

• • •

- □ 그런데 만약 학생 수가 갑자기 늘어난다면?
  - 변수를 추가로 만들어야 함

#### 배열이란 무엇인가?

- □ 변수가 여러 개 만들어질 때의 문제점
  - 관리의 어려움
  - 유사한 코드가 많아짐 (변수 정의, 값 넣기, 값 사용 등)
- 다수의 변수선언을 용이하게 하기 위해서 배열이라는 것이 제공됨. 배열을 이용하면 하나의 선언을 통해서 둘 이상의 변수를 선언할 수 있다.
- 배열은 단순히 다수의 변수 정의만 대신하지 않는다. 다수의 변수로는 할 수 없는 일을 배열을 이용하면 할 수 있음
- □ 배열은 1차원의 형태로도 2차원의 형태로도 만들수 있음. 여기서는 일단 1차원 형태의 배열에 대해서 학습

# 1차원 배열 정의에 필요한 것 세 가지

- □ 배열 정의 방법
  - 자료형 배열이름[요소의개수]

```
int oneDimArr[4];
```



생성되는 배열의 형태

- int 배열을 이루는 요소(변수)의 **자료형**
- oneDimArr 배열의 **이름**
- [4] 배열의 길이 (변수의 개수)

```
int arr1[7]; // 길이가 7인 int형 1차원 배열 arr1 float arr2[10]; // 길이가 10인 float형 1차원 배열 arr2 double arr3[12]; // 길이가 12인 double형 1차원 배열 arr3
```

#### 다양한 배열 선언의 예

# 정의된 1차원 배열의 접근

□ 배열의 요소는 일반 변수와 같음 1차원 배열 접근의 예

```
oneDimArr[0] = 10 /* 첫 번째 요소에 10을 저장 */
oneDimArr[1] = 12 /* 첫 번째 요소에 12을 저장 */
oneDimArr[2] = 25 /* 첫 번째 요소에 25을 저장 */
```



oneDimArr[idx] = 20; → "배열의 idx+1번째 요소에 20을 저장"

# 정의된 1차원 배열의 접근

```
int main(void) {
                        왼편의 예제를 통해서 느낄 수
 int arr[5];
                        있는 배열의 또 다른 매력은?
 int sum = 0, i;
                        반복을 통한 순차 접근
 arr[0] = 10; arr[1] = 20; arr[2] = 30;
 arr[3] = 40; arr[4] = 50;
 for (i = 0; i < 5; i++)
   sum += arr[i];
 printf("배열요소에 저장된 값의 합: %d\n", sum);
  return 0;
```

실행결 과

배열요소에 저장된 값의 합: 150

# 배열! 정의와 동시에 초기화하기

int arr1[5] = { 1, 2, 3, 4, 5 } 소기화 리스트로 초기화 출기화 결과



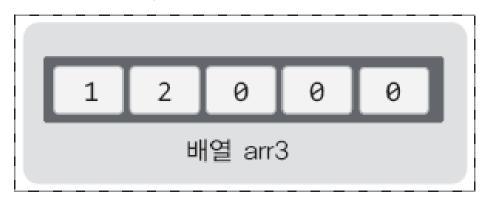
순서대로 초기화

# 배열! 정의와 동시에 초기화하기

int arr2[5] = { 1, 2 } 초기화 값 부족한 경우

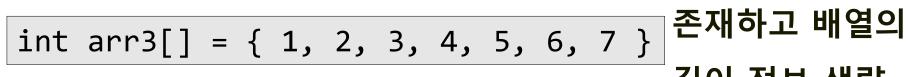


▶ 부족한 부분 0으로 채워짐



초기화 리스트는

길이 정보 생략





컴파일러가 배열의 길이 정보 채움

int  $arr3[7] = \{ 1, 2, 3, 4, 5, 6, 7 \}$ 

# 1차원 배열의 선언, 초기화 및 접근 관련 예제

```
int main(void) {
 int arr1[5] = \{ 1, 2, 3, 4, 5 \};
 int arr2[] = \{1, 2, 3, 4, 5, 6, 7\};
 int arr3[5] = { 1, 2 };
 int ar1Len, ar2Len, ar3Len, i;
 /* sizeof 연산의 결과로 배열의 바이트 크기정보 반환 */
 printf("배열 arr1의 크기:%d\n", sizeof(arr1));
 printf("배열 arr2의 크기:%d\n", sizeof(arr2));
 printf("배열 arr3의 크기:%d\n", sizeof(arr3));
```

# 1차원 배열의 선언, 초기화 및 접근 관련 예제

```
/* 배열의 길이를 계산하는 방식에 주목! */
ar1Len = sizeof(arr1) / sizeof(int);
ar2Len = sizeof(arr2) / sizeof(int);
ar3Len = sizeof(arr3) / sizeof(int);
/* 배열이기에 for문을 통한 순차적 접근이 가능하다 */
/* 다수의 변수라면 반복문을 통한 순차적 접근 불가능! */
for (i = 0; i < ar1Len; i++)
  printf("%d ", arr1[i]);
printf("\n");
for (i = 0; i < ar2Len; i++)
  printf("%d ", arr2[i]);
printf("\n");
```

# 1차원 배열의 선언, 초기화 및 접근 관련 예제

```
for (i = 0; i < ar3Len; i++)
 printf("%d ", arr3[i]);
printf("\n");
                       실행결과
return 0;
                       배열 arr1의 크기: 20
                       배열 arr2의 크기: 28
                       배열 arr3의 크기: 20
                       1 2 3 4 5
                       1 2 3 4 5 6 7
                       12000
```

#### char형 배열의 문자열 저장과 널 문자

char str[14] = "Good morning!"; 배열에 문자열 저장 자장 결과



문자열의 끝에 NULL 문자라 불리는 '\0'가 삽입되었음에 주목! NULL 문자는 문자열의 끝을 의미한다.

# char형 배열의 문자열 저장과 널 문자

```
int main(void) {
 char str[] = "Good morning!";
 printf("배열 str의 크기: %d\n", sizeof(str));
 printf("널 문자 문자형 출력: %c\n", str[13]));
 printf("널 문자 정수형 출력: %d\n", str[13]));
 str[12] = '?';
 printf("문자열 출력: %s\n", str);
                                 실행결과
 return 0;
                         배열 str의 크기: 14
                         널 문자 문자형 출력:
                         널 문자 정수형 출력: 0
```

문자열 출력: Good morning?

# 널 문자와 공백 문자의 비교

```
int main(void) {
  char nu = '\0';
  char sp = ' ';
  printf("%d %d\n", nu, sp);
  return 0;
}
```

- □ 널 문자를 %c를 이용해서 출력할 때 아무것도 나 타나지 않음 → 널 문자가 공백 문자는 아님
- □ 널 문자의 아스키 코드 값은 0이고, 공백 문자의 아스키 코드 값은 32임
- □ 널 문자는 모니터 출력에서 의미가 없어서 출력이 안될 뿐임

# scanf 함수를 이용한 문자열의 입력

```
int main(void) {
                               실행결과
 char str[50]; int idx = 0;
                            문자열 입력: Simple
 printf("문자열 입력: ");
                           입력 받은 문자열: Simple
                           문자 단위 출력: Simple
 scanf("%s", str);
 printf("입력 받은 문자열: %s\n", str);
 printf("문자 단위 출력: ");
 while (str[idx] != '\0') {
                          scanf 함수의 호출을
   printf("%c", str[idx]);
                          통해서 입력 받은
   idx++;
                          문자열의 끝에도 널
 printf("\n");
                          문자가 존재함을
 return 0;
                          확인하기 위한 문장
```

# scanf 함수를 이용한 문자열의 입력

- □ scanf 함수를 이용해서 문자열 입력 시 서식문자 %s를 사용
- □ 위와 같이 배열 이름 str의 앞에는 &(address of) 연산자를 붙이지 않음

```
char arr1[] = { 'H', 'i', '~' };
char arr2[] = { 'H', 'i', '~', '\0' };
```

- □ arr1은 문자열이 아닌 문자 배열, 반면 arr2는 문 자열!
- □ 널 문자의 존재여부는 문자열의 판단 여부가 됨

# 문자열의 끝에 널 문자가 필요한 이유

```
int main(void) {
  char str[50] = "I like C programming";
  printf("string: %s\n", str);
                                 □ 문자열의 시작은 판
                                   단할 수 있어도 문자
  str[8] = '\0';
                                   열의 끝은 판단이 불
  printf("string: %s\n", str);
                                   가능함
  str[6] = '\0';
                                 □ 따라서 문자열의 끝
  printf("string: %s\n", str);
                                   을 판단할 수 있도록
  str[1] = '\0';
                                   널 문자가 삽입됨
  printf("string: %s\n", str);
  return 0;
                                 string: I like C programming
                                 string: I like C
                                 string: I like
```

실행결과

string: I

# 문자열의 끝에 널 문자가 필요한 이유

□ 위 예제에서 보이듯이 printf 함수도 배열 str의 시작위치를 기준으로해서 널 문자를 만날 때까 지 출력을 진행한다. 따라서 널 문자가 없으면 printf 함수도 문자열의 끝을 알지 못한다.

# scanf 함수의 문자열 입력 특성

□ 앞에서 보인 예제를 실행할 때 다음과 같이 문자 열을 입력하면

He is my friend

□ 다음의 실행 결과를 보임

입력 받은 문자열: He

문자 단위 출력: He

- □ scanf 함수는 공백을 기준으로 데이터의 수를 구 분함
- □ 공백을 포함하는 문자열을 한 번의 scanf 함수 호출을 통해 읽지 못함 (이런 문자열 입력은 나중에 배움)

# 정리

- □ 배열의 이해
- □ 배열의 정의
- □ 배열의 초기화
- □ 문자 배열과 문자열