- string name[4] 라고 문자열 저장 공간 만들고

Name[0] = “emma”

Name[1] = “rodrigo”

Name[2] = “henry”

Name[3] = “kate”

로 초기화 시켜주면

name[0][0] 은 emma의 e를 말하는 것이다.

즉, 문자열의 배열 = 2차원 배열이다!!!!!!!!!!!!

<12. 문자와 문자열>

<12.1 문자와 문자열>

- ‘A’ : 하나의 문자

- “A” : 문자열. A와 \n이 저장되는 문자열.

<문자배열의 초기화>  
1. 초기값 나열하여 초기화 : 각 배열 원소 값 넣어주기 – 마지막 ‘\0’ 반드시 넣어주기

2. 문자열 상수로 초기화 예) “seoul”

3. 크기 지정 안하고 [] 쓰기 -> 초기화

<문자 입출력 라이브러리>

- getchar : char형 아니라 int형으로 반환한다. EOF문자(보통 -1로 정의)를 체크하기 위해서!

<문자열 입출력 라이브러리>

- scanf / printf : 한계 – 문자열이 띄어쓰기 있으면 거기서 끊김

- gets\_s : 줄바꿈 문자(엔터키) 나올 때 까지 한 줄의 라인을 문자열로 입력받는다.

<12.4 문자 처리 라이브러리>

- <ctype.h> 헤더파일에 정의

- 문자에 대한 검사 : 영문 알파벳인지, 숫자인지, 대문자인지, 소문자인지,

=> 함수 이름은 is…()로 시작

- 문자에 대한 변환 : 대문자를 소문자로 바꾸는 함수가 예시

=> 함수 이름은 to…()로 시작

<12.5 문자열 처리 라이브러리 함수>

1. 문자열 복사 : strcpy

2. 문자열 연결 : strcat

3. 문자열 비교 : strcpy

4. 문자 검색 : strchr

5. 문자열 검색 : char\* strstr

문자의 주소를 반환한다

6. 문자열 토큰 분리 : char\* strtok

부분 문자열의 주소를 반환한다

공백 문자로 분리된 단어를 예시로 들 수 있다

<12.6 문자열 수치 변환>

sscanf : 문자열을 수치로 변환

sprintf : 수치를 문자열로 변환

atoi : str형을 int형으로 변환

atof : str형을 double형으로 변환

<12.7 문자열 여러 개를 저장하는 방법>

- 2차원 문자 배열

=> char s[3][6] = {

“init”,

“open”,

“close”

};

- 문자열 길이가 다르면 2차원 배열에서 비는 공간 생기기 때문에, 메모리 낭비 막기 위해 포인터의 개념을 사용해야 한다.

=> char \*s[3] = {

“init”,

“open”,

“close”

};

=> 단점 : 문자열 상수를 만들어내는 것이기 때문에, 한번 결정되면 문자열의 내용을 변경할 수 없다.