**12장. 문자와 문자열**

**문자**: 하나의 글자

1. ‘ ‘ 를 이용해 표현
2. char형 자료형을 갖는다

**문자열**: 일련의 문자들의 모임

1. “ “ 를 이용해 표현

(큰 따옴표 안에만 있다면 문자 하나도 문자열이다. ex)”A”->문자열)

1. char형의 배열에 저장된다.
2. 문자열의 끝은 반드시 \0(NULL)로 끝나야 한다. (아스키코드 값=0)

* 문자열의 끝을 알려주는 것(이게 없으면 쓰레기 값이 들어가면서 문자열이 이상해질 수도..)

1. 배열의 인덱스를 통해서 접근, 변경 가능

**문자 배열의 초기화**

1. char str[4] = {‘a’, ’b’, ’c’, ’d’};

* 배열을 초기화 하듯이 넣어준다.

1. char str[6] = ”Hello”; / char str[ ] = ”Hello”;

* 문자열 상수를 사용해 초기화
* 배열 크기 지정 안 해주면 알아서 자동으로 배열 크기 설정

1. char str[ ] = “”; / char str[4] = “”;

* **빈 문자열(NULL)로 초기화**

**문자열의 출력**

1. 반복문을 통해서 하나씩 출력할 수 있음
2. %s 형식 지정자를 사용해 printf로 출력

**문자열의 변경**

1. 인덱스를 통해서 개별적으로 접근 가능

Ex) char str[ ] = “Hello”;

str[0] = ‘W’; -> Wello로 바뀜

(문자 하나하나를 바꾸는 것이므로 작은 따옴표 써야 한다!)

1. 라이브러리 함수인 strcpy()함수를 이용

Ex) strcpy(str, “World”);

1. 배열의 이름이 곧 포인터(배열의 주소 값을 저장)가 된다.

따라서 str = “World”; 이런 식으로 문자열상수를 대입하는 것은 배열의 주소에 문자열을 넣으려고 하는 어마무시한 일이다….(오류!!)

**문자열 상수와 포인터**

문자열 상수: “Hello World”와 같이 프로그램 소스 안에 포함된 문자열

상수이므로 수정이 불가능 하다. (텍스트 세그먼트에 저장된다.)

★ 텍스트 세그먼트: 값을 읽기만 하고 변경할 수 없는 메모리 영역

★ 데이터 세그먼트: 값을 변경할 수 있는 메모리 영역

1. 배열을 사용하는 문자열 상수

Char str [ ] = “Hello World”;

* 문자열 상수의 **값을(주소 아님)을 복사해서 자신의 배열에 붙여 넣음**

(메모리는 배열 메모리(데이터 세그먼트)이고 실질적인 값 “Hello World”만 가져다가 복.붙, 배열에서 수정, 변경이 가능)

//즉, 문자열 상수를 이용만 하는거지 문자열 상수의 주소를 갖는 것이 아님

(깊은 복사)

\*배열의 이름이 포인터 변수처럼 쓰인다고 해도 배열은 데이터 세그먼트이므로 변경, 수정 가능함. (텍스트 세그먼트와 헷갈리지 말자!!)

1. 포인터 변수를 사용하는 문자열 상수

Char \*p = “Hello World”;

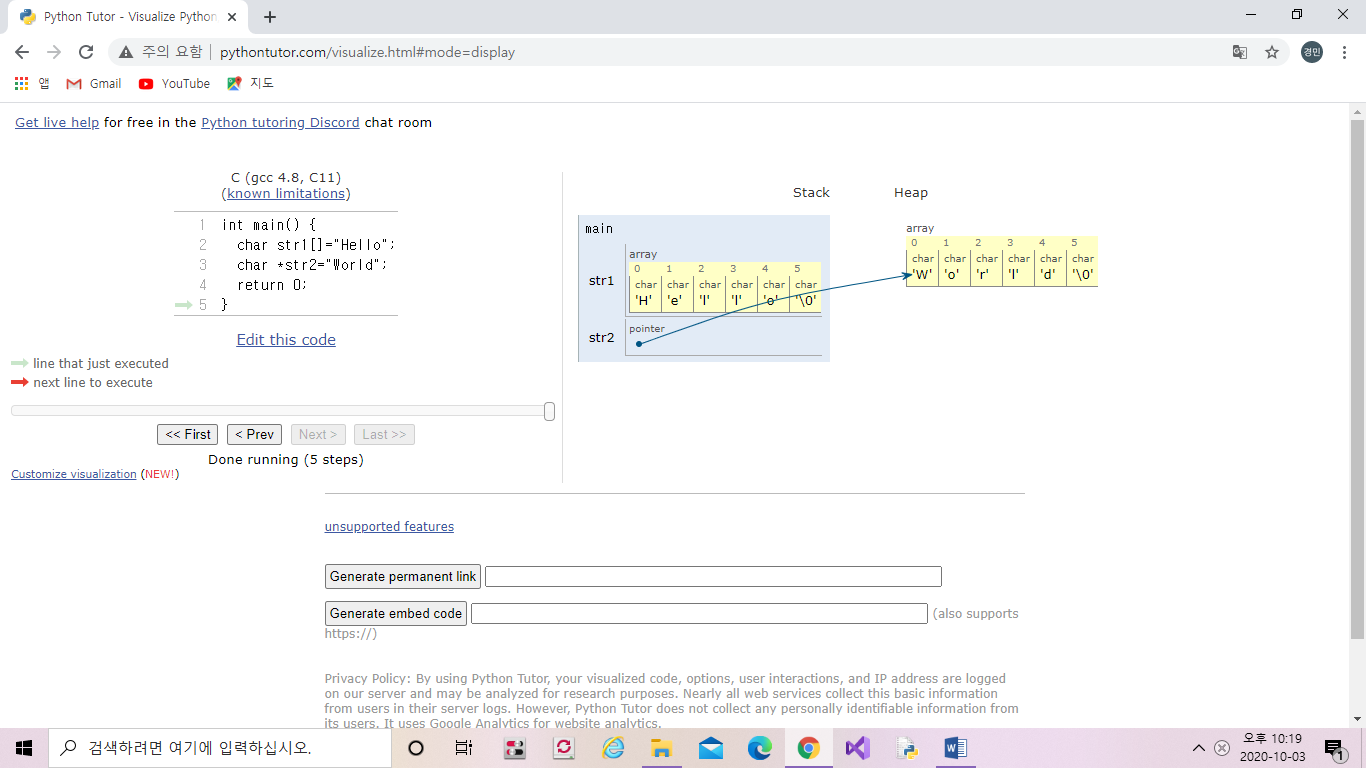
* 문자열 상수 “Hello World”의 주소 값이 포인터 변수 p에 저장

(p는 문자열 상수의 주소 값을 그대로 참조하므로 p에서 변경, 수정 불가)

(얕은 복사)

* P = “Good bye”; 문자열 변경을 위해선 새로운 문자열 상수를 참조해야 함

(포인트 ‘변수 자체’는 데이터 세그먼트 이므로 변수 p에 다른 문자열 상수(텍스트 세그먼트)를 참조



**문자 입출력 라이브러리**

|  |  |
| --- | --- |
| **int getchar(void)** | **하나의 문자를 읽어서 반환** |
| **void putchar(int c)** | **변수 c에 저장된 문자를 읽어서 출력** |
| **Int \_getch(void)** | **하나의 문자를 읽어서 반환 (버퍼x, 에코 x)** |
| **void \_putch(int c)** | **변수 c에 저장된 문자를 읽어서 출력 (버퍼x, 에코 x)** |
| **int \_getche(void)** | **하나의 문자를 읽어서 반환 (버퍼 x)** |
| **scanf(“%c”, &c)** | **하나의 문자를 읽어서 변수c에 저장한다.** |
| **printf(“%c”, c)** | **변수c에 저장된 문자를 출력한다.** |

: <stdio.h>

: <conio.h>, MS에서만 사용 가능

★ 버퍼

-> 문자를 입력하면 일단 버퍼(저장 공간)로 이동하고 엔터(\n)를 쳐야 프로그램으로

이동

★ EOF(End of File)

-> MS 콘솔에서는 Ctrl + Z / Linux 콘솔에서는 Ctrl + D

**문자열 입출력 라이브러리**

|  |  |
| --- | --- |
| **scanf(“%s”, s)** | **문자열을 읽어서 문자배열 s[]에 저장 / 스페이스 기준** |
| **printf(“%s”, s)** | **배열 s[]에 저장되어 있는 문자열을 출력한다.** |
| **char \*gets\_s(char \*s, int size)** | **한 줄의 문자열을 읽어서 문자 배열 s[]에 저장한다. / 엔터 기준** |
| **int puts(const char \*s)** | **배열 s[]에 저장되어 있는 한 줄의 문자열을 출력** |

★ char str[ ] =” Hello World”;

printf(%s, str) : %s는 배열 str에 저장된 문자열을 출력

printf(%u, str) : %u는 (unsigned int), 배열str의 주소값을 출력(배열의 이름은 포인터 역할을 하므로

str배열의 시작 주소값을 가르킴 즉, &(str[0])=str 이다.)

★ gets\_s() : \n(엔터가) 나올 때까지 입력 받고, \n은 \0으로 자동 변환되어 저장

(띄어쓰기 필요한 문자열은 이걸 사용!!)

puts(): \0은 \n으로 변경되고, 출력에 실패했을 경우, EOF가 반환

**문자 처리/변환 라이브러리**

★ 헤더파일 <ctype.h> 포함 필수

|  |  |
| --- | --- |
| **isalpha(c)** | **C가 영문자인가?** |
| **isupper(c)** | **C가 대문자인가?** |
| **islower(c)** | **C가 소문자인가?** |
| **isdigit(c)** | **C가 숫자인가?** |
| **isalnum(c)** | **C가 영문자나 숫자인가?** |
| **isxdigit(c)** | **C가 16진수 숫자인가?(0-9, A-F, a-f)** |
| **isspace(c)** | **C가 공백문자인가?(공백, \n,\t, \v, \r)** |
| **ispunct(c)** | **C가 구두점 문자인가?** |
| **isprint(c)** | **C가 출력 가능한 문자인가?** |
| **iscntrl(c)** | **C가 제어문자인가?** |
| **isascii(c)** | **C가 아스키 코드인가?** |
| **toupper(c)** | **C를 대문자로 바꿈( int형 인수 받고 int형 반환)** |
| **tolower(c)** | **C를 소문자로 바꿈( int형 인수 받고 int형 반환)** |
| **toascii(c)** | **C를 아스키 코드로 바꿈( int형 인수 받고 int형 반환)** |

**문자열 처리 라이브러리**

★헤더파일 <string.h> 포함 필수

|  |  |
| --- | --- |
| **strlen(s)** | **문자열 s의 길이를 구한다 / 원본 변경x** |
| **strcpy(s1,s2)** | **s2를 s1에 복사 / 완전 새로 복사** |
| **strcat(s1,s2)** | **s2를 s1의 끝에 붙여 넣음** |
| **strcmp(s1,s2)** | **s2과 s1을 비교 / s1에서 s2를 빼는 식으로 비교(숫자 반환)** |
| **strncpy(s1,s2,n)** | **s2의 최대 n개의 문자를 s1에 복사** |
| **strncat(s1,s2,n)** | **s2의 최대 n개의 문자를 s1의 끝에 붙여 넣음** |
| **strncmp(s1,s2,n)** | **최대 n개의 문자까지 s1과 s2를 비교** |
| **strchr(s,c)** | **문자열 s안에서 문자c를 찾아냄 / 찾은 주소 반환(가장 처음주소)** |
| **strstr(s1,s2)** | **문자열 s1에서 문자열 s2를 찾아냄 / 찾은 주소 반환(가장 처음 주소)** |
| **char \*strtok(s,delimit)** | **문자열 s를 delimit를 이용하여 토큰으로 분리** |

★strcmp(s1, s2) : s1에서 s2를 빼서 비교, 반환 값도 정수(아스키 코드 값을 비교)

-> s1-s2 < 0 이면, s2의 문자가 더 뒤에 있음 ex) s1=”abc” s2=”abd” / c(67) d(68)

-> s1-s2 > 0 이면, s2의 문자가 더 앞에 있음.

-> s1-s2 = 0이면, 두 문자열이 완전히 같음

★ **char\* strtok(s, delimit)** : **delimit(분리기준)도 문자열**이여야 한다.

Ex) 방법1) char ch[ ] = ” “;

strtok(s, ch);

방법2) strtok(s, “ “); or strtok(s, “\t”);

**계속해서 다음 토큰을 읽으려면 s 대신에 NULL을 넣어야 한다.**

Ex) char \*t1 = strtok(s, “ “); //첫 번째 토큰

char \*t2 = strtok(NULL, “ “); //두 번째 토큰

char \*t3 = strtok(NULL, “ “); //세 번째 토큰

**문자열 수치 변환**

문자열을 키보드나 화면이라고 생각한다….

1. **sscanf(s,%(변환한 수치의 형),변환한 수치를 저장할 변수의 주소) : 문자열->숫자**

(scanf는 키보드로 문자를 입력해 변수의 주소값에 저장, sscanf는 문자열을 입력 받았다고 생각하고 변수의 주소값에 저장)

1. **sprintf(s,%(수치의 형), 수치 값을 저장하고 있는 변수) : 숫자 -> 문자열**

(printf는 화면에 변수에 저장 된 값 출력, sprintf는 문자열에 저장 된 값을 출력)

ex)sprintf(s1, "Hello, %s", "world!");처럼 sprintf 함수에 문자열을 저장할 배열과 문자열을 만들 서식 그리고 문자열을 만들 값(문자열)을 순서대로 넣습니다. 이렇게 하면 "Hello, %s"에서 서식 지정자 %s 부분이 "world!"로 바뀌면서 s1에 "Hello, world!"가 저장됩니다

1. **int atoi(const char \*str) : str을 int로 반환**
2. **long atoi(const char \*str) : str을 long으로 반환**
3. **double atof(const char \*str) : str을 double로 반환**

**2차원 문자 배열**

char s[2][30] = {

**{“C program”} , //첫 번째 문자열//최대 30글자 저장 가능**

**{“JAVA”} //두 번째 문자열**

};

★ char s[행]은 포인터 역할을 한다.

**3차원 문자 배열**

**char dic[3][2][30] = {**

**{“book”, “책”},**

**{“boy”, “소년”},**

**{“girl”, “소녀”}**

**}; //두 가지의 정보(영어, 한글)을 30글자까지 저장할 수 있는 덩어리가 3개…**

★ 3차원 배열인 경우 char s[행][열]이 포인터가 된다.

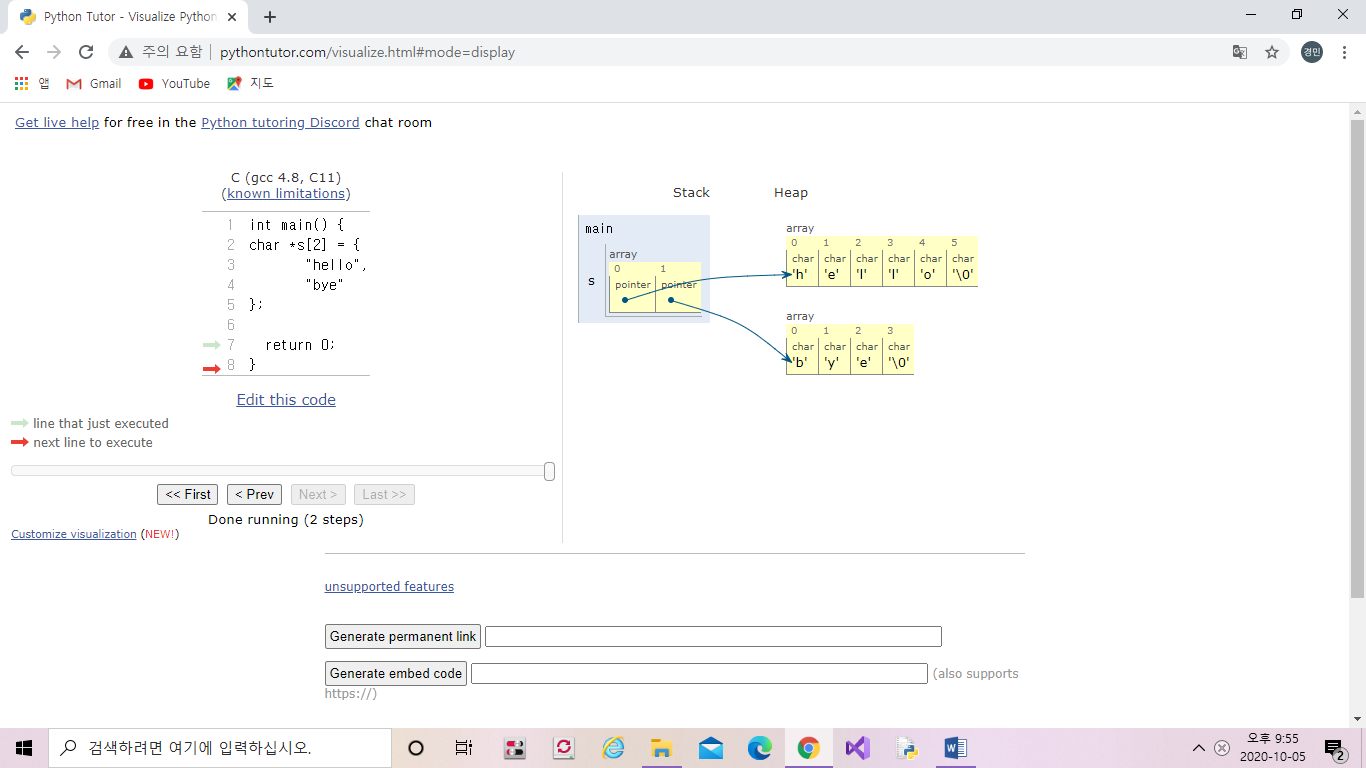
**문자 포인터 배열**

char \*s[행] = { //행만 있으면 된다….

str1,

str2

};



[**https://chuinggun.tistory.com/73?category=857702**](https://chuinggun.tistory.com/73?category=857702)