**C언어 입문**

#include<stdio.h>

int main(void) {

printf("Hello world");

}



위에 코드를 해석해보자.

#include<stdio.h>

먼저, 앞부분에 include는 영어 뜻과 같이 ‘포함하다’ 라는 의미한다. 그렇다면 무엇을 포함할까? stdio.h는 각각 std = standard, io = input/output, h = header 뜻을 가진 약자이다. 합치면 표준 입출력 헤더파일을 포함한다는 뜻이다. 지금단계에선 입력과 출력에 관련된 파일을 포함시켰구나 정도만 이해하면 된다. 다음부분을 확인해보면

int main(void) {

printf("Hello world");

}

바깥부분에 int main(void)은 메인 함수이다. 함수에 대한 부분은 함수파트에서 배우니 걱정하지 않아도 된다. c언어를 실행시키면 메인 함수에서부터 시작합니다. 그렇기에, 우리가 앞으로 계속 쓸 함수입니다. 중괄호”{ }”는 main함수의 영역을 표시한 것이다. 중괄호를 통해 컴퓨터는 함수의 처음과 끝을 알 수 있다. 메인 함수안에 Printf가 있다. Printf는 아까 포함했던 헤더파일에 있는 대표적인 출력 함수이다. 큰 따옴표 안에 있는 문자들을 출력한다.

이제 위의 코드를 해석할 수 있다.

**변수**

변수란 값(데이터)를 저장하기 위한 메모리 공간이다. 변수를 사용해주기 위해선, 선언하여 컴퓨터에게 변수의 사용을 알려줘야 한다. 선언을 해주면 컴퓨터는 해당 변수의 공간을 마련한다.

변수 a, b를 선언했다 가정해보자.

컴퓨터는 301호를 a의 집으로, 502호를 b의 집으로 제공한다.

a에 5를 대입하면(a = 5), 301호에 5가 저장된다. 이제 우리는 저장된 값을 이용하면 된다.

a의 값을 사용할 때는 301호안에서 값을 가져온다.

선언을 할 때, 변수의 형태를 정해줘야 한다. 각각의 형태는 구분되며 실수형태로 선언된 변수는 정수형태로 쓰이지 못한다. 따라서 변수의 자료형 선택에 주의해야한다.

변수의 형태는 크게 **문자형(char), 정수형 (int), 실수형(float, double)**이 있다.

**변수 이름 규칙**

1. 변수의 이름은 영문자(대소문자), 숫자, 언더스코어(\_)로만 구성한다.

ex) int 숫자 -> 불가능(한글사용불가)

2. 변수의 이름은 숫자로 시작될 수 없다.

ex) int 3\_num -> 불가능

3. 변수의 이름 사이에는 공백을 포함할 수 없다.

4. 변수의 이름으로 C언어에서 미리 정의되어 특수목적으로 사용하는 키워드(keyword)는 사용할 수 없다.

ex)int, char, double, printf, scanf 등은 변수이름으로 사용할 수 없다.

또한, 선언된 변수의 개수가 많아질수록, 기존의 변수들과 헷갈리기에 변수의 이름을 정해줄 때 의미를 부여하는 것이 좋다.

**출력함수 Printf**

이제 대표적인 출력함수 printf 함수를 배워보자.

Printf는 stdio.h에 있는 대표적인 출력함수이며, 함수의 인자로 오는 문자열을 출력한다.

형식은 간단하다. printf("내가 출력하고 싶은 문자열");

큰 따옴표안에 내가 출력하고 싶은 것을 출력하면 된다.

Printf(‘A’) 같은 경우 error가 발생한다. 작은 따옴표는 하나의 문자를 의미하기에, 문자열만 취급하는 printf와 호환이 불가하다.

#include<stdio.h>

int main(void)

{

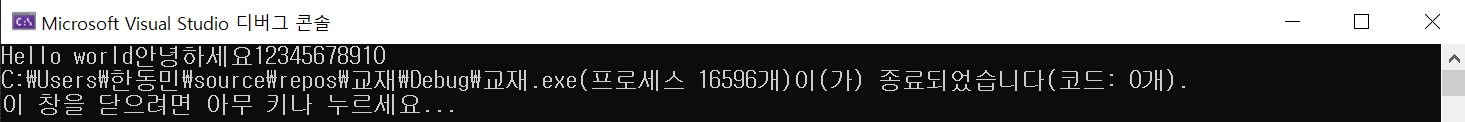
//printf('Hi'); error c언어에서는 문자열은 ""로 표기하고 문자는 ''로 표기한다.

printf("Hello world");

printf("안녕하세요");

printf("12345678910");

}



위에 내용을 코드를 출력하면 다음과 같이 한 줄로 출력된다. 줄을 바꾸려면 어떻게 해야 될까?

테이블이(가) 표시된 사진

자동 생성된 설명**<이스케이프 시퀀스>**

c언어 출력에서 편의기능을 제공하는 것이 이스케이프 시퀀스이다.이스케이프 시퀀스는 모두 앞에 '\'가 입력된다.

\n : newline(줄바꿈), 다음으로 출력할 부분에 줄을 바꿔준다.

\t : 키보드에 있는 TAP 기능

“\n” 은 자주 쓰는 이스케이프 시퀀스이니, 꼭 기억하길 바란다.

#include<stdio.h>

int main(void)

{

printf("\n");

printf("Hello world\n");

printf("안녕하세요\n");

printf("12345678910\n");

printf("\tHello world \"Hello world\" \n");

}

텍스트이(가) 표시된 사진

자동 생성된 설명

**<형식지정자>**

우리가 단순하게 입력한 것만 출력한다면, 프로그램을 만드는 이유가 없다. 컴퓨터 내부에서 을 한 결과를 출력해야 의미가 있다. 앞에서 배운 변수를 Printf 함수를 통해 출력해보자.

Printf에서 변수를 출력하기위해서 출력형식을 정해줘야 한다. 같은 변수(=값)이어도 출력형식이 다르다면 출력은 다르다.

테이블이(가) 표시된 사진

자동 생성된 설명

#include<stdio.h>

int main(void)

{

int n = 5;

char c = 'A';

double f = 3.14;

printf("숫자출력 %d\n", 10);

printf("정수형 : %d, 실수형 : %f, 문자형 : %c\n", n, f, c);

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

앞에서 배운 형식지정자와, 이스케이프 시퀀스 중 “\n”을 이용해 아래와 같이 출력해보자.

===========================

이름 : 김세영

혈액형 : A형

키 : 165cm

몸무게 : 50.6kg

===========================

#include<stdio.h>

int main(void)

{

char o = 'a';

int m = 165;

double k = 50.6;

printf("이름 : 김세영\n");

printf("혈액형 : %c\n", o);

printf("키 : %dcm\n", m);

printf("몸무게 : %0.2fkg\n", k);

//위의 코드처럼 실수형을 출력할 때는 소수점 자리수를 정해줄 수 있다.

//%0.원하는 소수점자리f

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

**입력함수 scanf**

대표적인 출력함수를 배웠으니 입력함수를 배워보자. scanf 함수는 stdio.h에 있는 대표적인 입력함수이다. printf 함수에서는 출력했던 큰 따옴표안이 입력형식이 된다. Scanf 또한, 형식지정자(ex. %d)를 통해 입력 받는다.

예시를 통해 확인해보자.

scanf("%d %c %lf", &변수1, &변수2, &변수3); 라는 코드가 있다해보자.

위의 예에서는 (정수형 변수)(공백)(문자형 변수)(공백)(실수형 변수)형식으로 입력을 해줘야 한다.

입력 받은 값은 변수1, 변수2, 변수3에 순서대로 들어간다.

아래의 코드를 보면서 이해해보자.

#include<stdio.h>

int main(void)

{

char var1;

int var2;

float var3;

double var4;

scanf("%c %d %f %lf", &var1, &var2, &var3, &var4);

//(문자)(공백)(정수)(공백)(실수)(공백)(실수) 형식으로 입력을 해줘야한다.

printf("\n\n출력결과 :: %c %d %f %lf\n", var1, var2, var3, var4);

}

텍스트이(가) 표시된 사진

자동 생성된 설명

**<주소 연산자(&)>**

주소연산자는 해당 변수의 주소를 연산해준다.

&가 없다면, “변수의 주소”가 아닌 “변수가 가진 값”으로 인식하기에 &를 붙여야 한다.

#보충설명

c언어에서 변수는 '=' 왼쪽에 있는 경우를 제외하면 컴퓨터는 해당 변수를 값으로 여긴다.

Scanf 함수에서 주소연산자를 사용하지 않으면, 변수의 값이 가리키는 주소에 입력을 받게 된다. (엉뚱한 곳에 입력된다.)

앞서 변수를 배울 때 a에게 301호의 방을 주고, b에게 302호의 방을 준다는 예를 들었다.

a에 201을 대입한 뒤, scanf("%d", a); 와 같이 주소연산자를 생략했다고 해보자, 그러면 입력 받은 숫자는 a의 방인 301호가 아닌, a의 값인 201호로 들어간다.

누가 쓰는 지도 모르는 엉뚱한 방에 들어간 것이다. 그러므로 주소연산자를 꼭 사용해야 한다.

아래의 코드를 통해 주소연산자의 유무를 확인해보자.

#include<stdio.h>

int main(void)

{

char var1;

int var2 = 100000;

printf("값 : %d, 주소 : %p\n", var2, &var2);

//%p는 해당 주소형태를 출력하는 형식지정자

}



**조건문**

기본적인 입출력 함수들을 배웠으니, 흐름을 제어하는 명령어를 배울 것이다.

먼저, 예를 들어 옷을 선택해주는 프로그램이 있다하자 내가 어느 장소에 갈지에 따라서 옷의 종류는 달라진다. 학교에 갈 때는 교복, 운동 갈 때는 추리닝, 결혼식장을 갈 때는 단정한 옷 등 각각의 장소에 따라 구분해야 한다.

이처럼 컴퓨터에서도 조건에 따라 구분이 필요하다.

c언에서 조건문은 if이다.

if(실행할 조건){

소괄호안에 있는 조건이 참이라면 if의 중괄호안에 있는 명령어를 실행시킨다.

}

else{

if의 조건이 거짓이라면 해당 중괄호안에 있는 명령어를 실행시킨다.

}

이제 조건식에 들어갈 비교연산자에 대해 알아보자.

**<조건식을 구성하는 비교 연산자>**

비교연산자란 상대적인 크기를 판단하는 연산자이다.

테이블이(가) 표시된 사진

자동 생성된 설명

컴퓨터는 하나의 비교만 할 수 있기에, 2개 이상의 비교를 하려면 논리연산자를 써야한다.

논리연산자란 주어진 논리식을 판단하여 참과 거짓을 반환한다.

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

이제, 문제를 보며, if – else 를 사용해보자.

문제1) 빈 곳을 채워, 계산기프로그램을 완성하시오.

예제 입력 : 2 \* 3

예제 출력 : 6

#include<stdio.h>

int main() {

int n1, n2;

char op;

scanf("%d %c %d", &n1, &op, &n2);

if () {//덧셈

printf("결과값은 : %d입니다.\n", n1 + n2);

}

if () {//뺄셈

}

if () {//곱셈

}

}

**<정답>**

#include<stdio.h>

int main()

{

int n1, n2;

char op;

scanf("%d %c %d", &n1, &op, &n2);

if (op == '+') {//덧셈

printf("결과값은 : %d입니다.\n", n1 + n2);

}

if (op == '-') {//뺄셈

printf("결과값은 : %d입니다.\n", n1 - n2);

}

if (op == '\*') {//곱셈

printf("결과값은 : %d입니다.\n", n1 \* n2);

}

return 0;

}

문제2) 계산기 프로그램 확장(1번 문제의 확장)

나누기 연산까지 제공하는 계산기를 제작하시오.

Ex) 3 / 2 -> 1.5 (정수형의 연산으로는 1.5가 아닌 1이 나옴)

**<정답>**

#include<stdio.h>

int main()

{

int n1, n2;

char op;

scanf("%d %c %d", &n1, &op, &n2);

if (op == '+') {//덧셈

printf("결과값은 : %d입니다.\n", n1 + n2);

}

if (op == '-') {//뺄셈

printf("결과값은 : %d입니다.\n", n1 - n2);

}

if (op == '\*') {//곱셈

printf("결과값은 : %d입니다.\n", n1 \* n2);

}

if (op == '/') {

printf("결과값은 : %f입니다.\n", double(n1) / n2);

}

return 0;

}

문제3) 숫자를 입력받아 1, 10, 100의 자릿수의 합을 더한 결과를 출력하시오

(힌트 : /와 %연산자를 이용)`

예시 입력 : 123

출력 : 6

**<정답>**

#include<stdio.h>

int main()

{

int num, sum = 0;

scanf("%d", &num);

if (num) {

sum += num % 10;

}

num /= 10;

if (num) {

sum += num % 10;

}

num /= 10;

if (num) {

sum += num % 10;

}

printf("%d\n", sum);

return 0;

}

**<연산자 약자.>**

sum++

sum = sum + 1

sum += 1.

이렇게 3개의 식은 모두 sum에다가 sum + 1을 대입한다는 의미이다.

전위연산자( ++var) : 명령어를 실행시키기 전에 연산을 진행

후위연산자( var++) : 명령어를 실행시킨 후 연산을 진행

소스코드를 보면서 연산자에 대해 알아보자.

#include<stdio.h>

int main()

{

int a = 5;

printf("====================\n");

printf("%d\n", a++);//후위 연산자

printf("%d\n", a);

printf("%d\n", ++a);//전위 연산자

printf("====================\n");

int b = 3;

b += 3;

printf("%d\n", b);

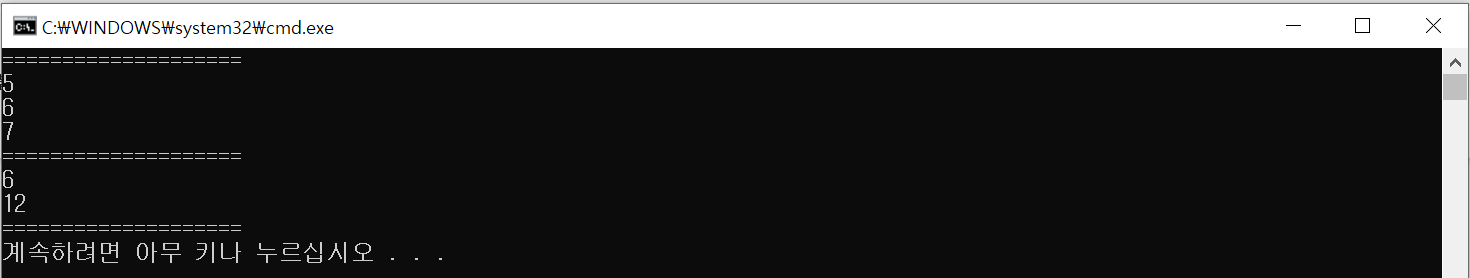
b \*= 2;

printf("%d\n", b);

printf("====================\n");

return 0;

}



**반복문**

위에 반복문을 보다 보면, 반복적인 형태의 명령어가 많다. 이런 코드들을 간편하게 쓰기위해 반복문이 만들어졌다.

반복문은 같은 명령어를 일정한 횟수만큼 반복적으로 수행시키는 명령어다.

예를 들어 3의 20승을 구하고 싶다면, 3을 곱하는 명령어를 20번 수행시키면 된다.

반복문을 쓰지 않는다면, 20줄의 명령어를 작성해야 하지만, 반복문을 사용하면 단 3줄이면 된다.

**<반복문 종류>**

1. while , 2. do-while, 3. for

for은 1,2보다 간결하게 반복문의 코드를 작성할 수 있기에 우리는 for을 자주 사용할 것이다.

**for(초기값 ; 반복조건 ; 증감식){**

**~반복할 명령어들~**

**}**

While(반복조건){

~명령어~

}

Do{

~명령어~

}while(반복조건)

for을 정확히 이해하기 위해선 반복하는 순서에 대해 알아야 한다.

**<for의 실행순서 >**

Ex) for(i=0;i<5;i++){

Printf(“\*”);

}

1) 반복문을 실행하면 초기값에 대한 명령어를 수행한다. Ex) i=0

2) 반복조건이 만족했는 지 확인한다. Ex) i<5 :: i가 5보다 작은 지 확인

3) 조건을 만족하지 못하면, 반복문이 끝나고, 만족하면 내부에 있는 명령어를 모두 실행한다.

Ex) printf(“\*”);

4) 반복문에 증감식에 있는 명령어를 수행한다. Ex) i++

5) 반복조건이 참인지 검사한다. 거짓이면 반복문을 끝낸다. 거짓이 아니면 6번과정으로 넘어간다.

6) 반복문안에 있는 명령어를 모두 실행한다. Ex) printf(“\*”);

5, 6번 과정을 계속 반복한다.

**<예제>**

#include<stdio.h>

int main()

{

int i, a, b, sum = 0;//sum은 초기화

for (i = 1; i <= 10; i++)

{

printf("%d번째, 안녕하세요 \n", i);

}

for (i = 1; i <= 10; i++)//1~10까지의 합을 구하는 방법

{

sum = sum + i;

}

printf("1~10까지의 합은 : %d\n", sum);

}

텍스트이(가) 표시된 사진

자동 생성된 설명

문제1) 1~100까지의 짝수의 합을 구하시오.

**<정답>**

#include<stdio.h>

int main()

{

int i, sum = 0;

for (i = 1; i <= 100; i++) {

if (i % 2 == 0) {

sum += i;

}

}

printf("%d", sum);

return 0;

}

문제2) 숫자 하나를 입력받아 1~N까지의 짝수의 합을 출력하시오.

예시 입력 : 10

출력 : 30

**<정답>**

#include<stdio.h>

int main()

{

int i, n, sum = 0;

scanf("%d", &n);

for (i = 1; i <= n;i++) {

sum += i;

}

printf("%d", sum);

return 0;

}

문제3) 숫자를 하나 입력받아, 2^n을 구하시오.

예시입력 : 5

출력 : 32

**<정답>**

#include<stdio.h>

int main()

{

int n, i, result = 1;

scanf("%d", &n);

for (i = 1; i <= n; i++) {

result = result \* 2;

}

printf("%d", result);

}

문제4) 숫자를 하나 입력받아, 약수의 개수를 구하시오.

예시 입력 : 10

출력 : 4

**<정답>**

#include<stdio.h>

int main()

{

int n, i, count = 0;

scanf("%d", &n);

for (i = 2; i <= n; i++) {

if (n % i == 0) {

count++;

}

}

printf("%d", count);

}

문제5) 2~100 소수의 합을 구하시오.

**<정답>**

#include<stdio.h>

int main()

{

int i, n, count, sum = 0;

for (i = 2;i <= 100;i++) {

count = 0;

for (int j = 1; j <= i; j++) {

if (i % j == 0) {

count++;

}

}

if (count == 2) {

sum += i;

}

}

printf("%d", sum);

}

**배열(array)**

배열이란 같은 자료형으로 이루어진 집합의 형태로 각 요소들은 첫번째 요소에 대한 상대적 거리로 표현된다. 배열을 사용함으로 다수의 변수들을 하나의 단위로 묶을 수 있고, 각 요소에 대한 접근이 용이하다.

예를 들면) 1학년 1반의 30명의 수학성적은 “수학성적”이라는 같은 자료형으로 이루어진 집합 형태이므로 배열로 표현할 수 있다. Ex) int score[30] = {100, 90, 65 , 82, ~ , 88};

“score[3] == 82”에서 대괄호안에 있는 숫자를 인덱스라 한다. 인덱스는 배열에서 상대적인 위치를 나타낸다. 또한, 배열의 시작은 ‘0’부터 시작하므로 인덱스는 0부터 존재한다.

위의 성적에 대해 더 설명하면, 각각의 점수들을 배열의 요소이다. 각 점수들은 1번 학생 성적의 위치는 score[0], 2번학생 성적의 위치는 score[1], n번학생 성적의 위치는 score[n-1]에 존재한다.

배열의 선언형식은 (자료형)(배열이름)([배열의크기]) 이다.

Ex) int arr[5];는 정수형 배열로 크기가 5이며 이름이 arr인 배열이다.

**<주의할 점>**

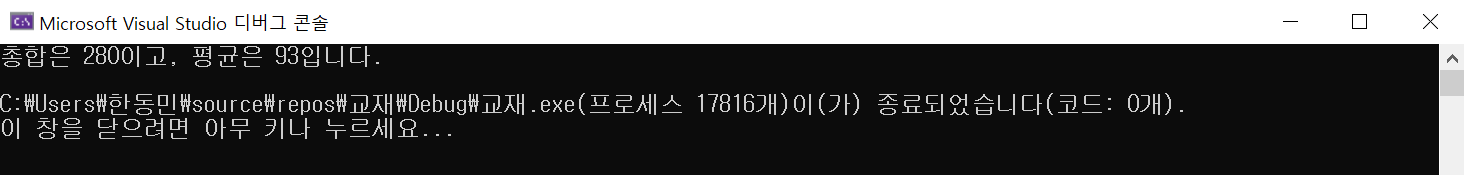
int a = 5;

int arr[a]; // error 변수가 아닌 상수를 통해 크기를 선언해야 한다.

int arr[5]; // 가능

**<예제 코드>**

ㄴ



문제1) 숫자 3개를 받고, 입력받은 숫자를 3번씩 출력하시오

예시 입력: 1 2 3

예시 출력: 1 2 3

1 2 3

1 2 3

#include<stdio.h>

int main() {

int arr[3];

for (int i = 0; i < 3; i++) {

scanf("%d", &arr[i]);

}

for (int i = 0; i < 3;i++) {

for (int j = 0;j < 3;j++) {

printf("%d ", arr[j]);

}

printf("\n");

}

}

문제2) 학생 10명의 키를 입력받고, 170이상의 키를 출력하시오.

예시 입력: 150 155 160 165 168 169 170 175 180 155

예시 출력: 170 175 180

문제3) 학생 10명의 성적을 입력받고, 1등한 학생의 입력순서와 점수를 출력하시오.

예시 입력: 80 100 92 98 76 96 88 96 76 82

예시 출력: 2번 학생이 100점으로 1등

#include<stdio.h>

int main() {

int i, m, max = 0, arr[10];

for (i = 0;i < 10;i++) {

scanf("%d", &arr[i]);

}

for (i = 0;i < 10;i++) {

if (arr[max] < arr[i]) {

max = i;

}

}

printf("%d번 학생이 %d점으로 1등", max + 1, arr[max]);

}

**문자열**

컴퓨터에서는 모든 것이 0과 1 로 구성되었다. 그렇기에 컴퓨터가 알아들을 수 있는 것은 2진수, 숫자만 알아들을 수 있다. 컴퓨터가 문자 자료형을 사용하기 위해 문자와 숫자를 각각 일대일로 대응시키는 방법을 고안했다. 대표적으로 아스키코드, 유니코드, UTF-8이 있다.

테이블이(가) 표시된 사진

자동 생성된 설명C언어에서는 아스키코드를 채택하여 사용하고 있으므로 우리는 문자형을 다룰 때 아스키코드를 참고한다. 오른쪽은 아스키코드의 일부분이다.

예시) 문자형 변수에 숫자값이 65라면 ‘A’를 가리키는 것이고, 33이면 ‘!’를 가리키는 것이다.

아래의 예시코드를 통해 이해해보자.

#include<stdio.h>

int main() {

char c1 = '!';

char c2 = 65;

printf("%c에 대응되는 숫자 : %d\n", c1, c1);

printf("%c에 대응되는 숫자 : %d\n", c2, c2);

//아스키코드를 보면 소문자와 대문자의 값의 차이는 32이다. 이를 통해 대소문자 변환을 할 수 있다.

printf("%c의 대문자는 %c이다.\n", c2, c2 + 32);

}

텍스트이(가) 표시된 사진

자동 생성된 설명

자 이제, 문자열이란 무엇일 까? 문자열은 말 그대로 연속된 문자들의 집합이다. 영어로는 “String”이라고 한다. “AB”도 A와 B가 연속된 문자열이다. C언어에서는 문자열의 끝을 파악하기 위해 문자열의 마지막에 '\0' 이라는 NULL문자를 넣는다. 해당 문자는 문자열의 끝을 알려준다.

문자열을 위한 여러 형식이 존재한다. 우리가 앞에서 배운 Print, sacnf에서도 문자열을 위한 형식지정자가 존재한다. ex) printf(“%s”, string), sacnf(“%s,string)

#include<stdio.h>

int main() {

char string[9] = "aBcDeFgH";

printf("%s\n", string);

string[5] = '\0'; //문자열의 끝을 변경

printf("%s\n", string);

//해당값을 모두 대문자로 출력할 수 있다.

for (int i = 0; i < 9; i++) {

if (96 < string[i] && string[i] < 123) {

string[i] -= 32;

}

printf("%c", string[i]);

}

}

텍스트이(가) 표시된 사진

자동 생성된 설명

**2차원 배열**

2차원 배열은 배열을 단순히 2차원화 한 것이다. 지도, 좌표에 대한 표현 등 2차원 배열을 사용하면 데이터를 직관적으로 표현할 수 있다. 2차원 배열은 행과 열을 통해 표현된다.

예를 들어 arr[1][2]가 있다고 하자.

행은 가로줄을 나타내고 열은 세로줄을 나타낸다. 엑셀화면을 통해 봤던 행렬의 개념과 같다.

arr[1][2]의 위치에 대해 말하면 배열은 0부터 시작하니까 2행 3열의 값을 의미한다.

1 2 3 4

5 6 7 8

배열에 위와 같이 저장되었다면 arr[1][2]는 7을 의미한다.

#include<stdio.h>

int main() {

//4명의 학생의 국어, 영어, 수학성적에 대한 배열

//행은 학생을 의미하고, 열은 각 성적을 의미한다.

int score[4][3] = {

{100, 90 ,96}, //첫번째학생의 성적

{88, 92, 76},

{100, 80, 98},

{90, 76, 92}

};

char name[3][10] = {

"국어",

"영어",

"수학"

};

int sum;

for (int i = 0; i < 3;i++) {

sum = 0;

for (int j = 0; j < 4; j++) {

sum += score[j][i];

}

printf("%s 성적의 총합 : %d\n", name[i], sum);

}

}

텍스트이(가) 표시된 사진

자동 생성된 설명

**<예제 : 2차원 배열 순서대로 채우기>**

#include<stdio.h>

int main() {

int n;

int arr[100][100];

scanf("%d", &n);

for (int i = 0, count = 1; i < n;i++) {

for (int j = 0; j < n;j++) {

arr[i][j] = count++;

}

}

for (int i = 0; i < n;i++) {

for (int j = 0; j < n;j++) {

printf("%d ", arr[i][j]);

}

printf("\n");

}

}

텍스트이(가) 표시된 사진

자동 생성된 설명

**함수**

**영역(scope)**

c언어에서 영역을 중괄호를 통해 구분한다. 해당 영역에서만 사용하는 변수를 지역변수라고 한다. 영역을 벗어나면 변수를 사용할 수 없다. 모든 영역에서 사용이 가능한 변수를 전역변수라고 한다. 괄호밖에 선언된 변수이다. 지역변수는 해당 영역을 벗어나거나, 함수가 끝나는 등 쓰임이 다하면 쓸 수 없지만, 전역변수는 프로그램이 종료될 때까지 쓸 수 있다.

아래의 코드를 확인해보자.

#include<stdio.h>

int global = 5; // 전역변수

int sum() {

int local = 1;//지역변수

return local;

}

int main()

{

{

int local = 2;//지역변수

global++;

printf("지역변수 : %d, 전역변수 : %d\n", local, global);

}

{

//지역변수, 앞에 있는 local는 해당 괄호에서만 유효하므로 이름은 같지만, 다른 변수

int local = 3;

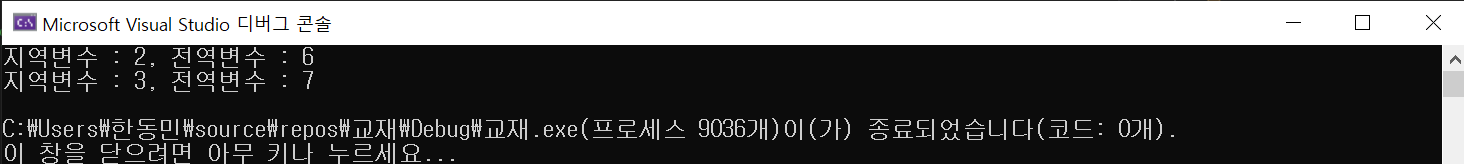
global++;

printf("지역변수 : %d, 전역변수 : %d\n", local, global);

}

//printf("%d\n", local); error //해당 영역에서 선언된 a라는 변수가 없기때문에

}



함수란

어떤 목적을 위해 독립적으로 작성된 코드의 집합을 의미한다. 주로 반복적인 작업이 필요할 때, 사용한다. 함수를 사용하기 위해선 먼저 함수를 선언해야 한다. 그 후 작업이 필요한 공간에서 함수를 호출하면, 함수를 사용할 수 있다.

함수 선언 형식은 아래와 같다.

(반환형)(함수이름)(입력인자){

~~~명령어~~~

}

int sum(int a, int b) { //정수형 변수 2개를 입력받아, 두 수의 합을 반환하는 함수이다.

return a + b;

}

반환형은 우리가 알고 있는 기존의 자료형(int, char, double 등)이외에 void형이 있다.

void형은 텅빈 자료형으로 아무것도 return하지않는다고 생각하면 된다.

**<예제>**

#include<stdio.h>

//(반환형 : 정수형)(함수이름 : sum)(입력 : 정수형변수 2개)

//두 수의 합을 구하는 함수

int sum(int a, int b) {

return a + b;

}

//n의 m승을 구하는 함수

int pow(int n, int m) {

int temp = 1;

for (int i = 0; i < m; i++) {

temp \*= n;

}

return temp;

}

int main()

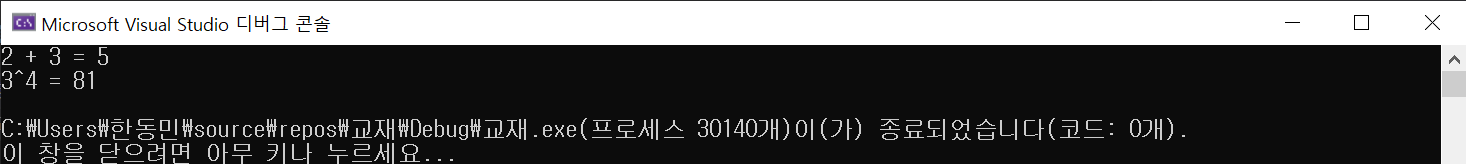
{

printf("%d + %d = %d \n", 2, 3, sum(2,3));

printf("%d^%d = %d \n", 3, 4, pow(3, 4));

return 0;

}



함수 중에 특별한 함수가 있다. 바로 재귀함수이다.

재귀함수란 함수 선언부에서 자신을 호출하는 함수를 의미한다. 엘리베이터안에 있는 거울을 보면 거울속에 작은 거울이 반복적으로 있는 것을 확인할 수 있다. 재귀함수 또한 반복적인 표현을 통해 문제를 해결하는 함수이다.

주의할 점

1)함수는 다른 시간, 다른 입력이면 다른 함수이다.

2)함수에 input을 전달할 때, 값을 복사하는 방식으로 전달한다.

#include<stdio.h>

int n\_sum(int a)

{

if (a == 1)

return 1;

return a + n\_sum(a - 1);

}

void prt1(int a)

{

if (a != 0){

printf("%d ", a);

prt1(a - 1);

}

}

void prt2(int a)

{

if (a != 0) {

prt2(a - 1);

printf("%d ", a);

}

}

int main()

{

printf("prt1 함수의 결과 : ");

prt1(5);

printf("\nprt2 함수의 결과 : ");

prt2(5);

printf("\n1 ~ 10 까지의 합은 : %d", n\_sum(10));

return 0;

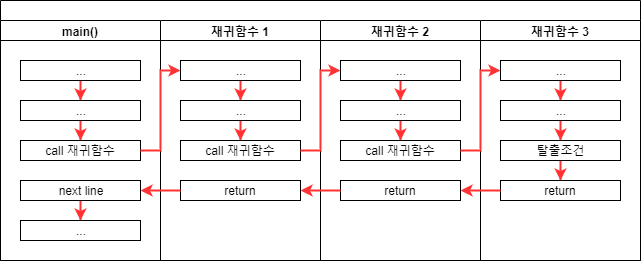
}

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 prt1, prt2 두 함수가 결과가 다른 것을 이해하기 힘들다면, 아래의 순서도를 보자.

재귀함수의 순서도



출처<[**10분 TV**](http://10bun.tv/)>

해당 순서도를 통해 위의 코드를 설명하면, prt1 함수는 출력을 한 뒤, 재귀호출하기에 내림차순으로 출력된다. 반면 prt2 함수는 재귀호출을 한 뒤 출력을 하기에 오름차순으로 출력된다.

아래의 문제들과 풀이코드들을 보며 재귀함수에 대해 깊이 이해해보자.

문제1) 입력받은 수까지의 곱을 반환하는 재귀함수를 작성하시오.

예시 입력 : 5

예시 출력 : 120

int f(int a) {

if (a < 1) {

return 1;

}

return a \* f(a - 1);

}

문제2) n \* n의 정사각형을 출력하시오

예시 입력 : 3

예시 출력 : \*\*\*

\*\*\*

\*\*\*

void f1(int a) {

if(a < 1){

return;

}

f2(a);

printf("\n");

f1(a - 1);

}

void f2(int a) {

if (a > 0) {

printf("\*");

f2(a - 1);

}

}

문제3) 이등변삼각형을 출력하시오

입력: 3

출력: \*

\*\*\*

\*\*\*\*\*

void f1(int a, int b) {

if (a > b) {

return;

}

f2(2\*a - 1);

printf("\n");

f1(a+1, b);

}

void f2(int a) {

if (a > 0) {

printf("\*");

f2(a - 1);

}

}

문제4) n번째 항의 피보나치 수를 구하시오.

예시 입력: 6

예시 출력: 8

int f(int n) {

if (n <= 2) {

return 1;

}

return f(n - 1) + f(n - 2);

}

**구조체**

작성자가 새로운 자료형을 정의할 수 있다.

학생관리프로그램을 제작한다면, 학생에 대한 자료형을 만들어야 할 것이다.

학생이라는 자료형에는 학생의 이름, 학번, 전공이 들어갈 것이다.

struct student {

char major[10];

char name[10];

int id;

};

이와 같이 c언어에서 기본으로 제공하는 자료형(int, double, char 등)을 이용하여 새로운 타입을 정의할 수 있다.

구조체안에 있는 변수들은 구조체의 멤버변수라고 한다.

학생구조체의 멤버변수로는 길이가 10인 문자열 major, 정수형 student\_id, student\_grades가 있다. 멤버변수에 접근을 할 때는 멤버연산자 '.'를 사용한다.

ex) 학생자료형의 멤버변수 학번의 값을 출력한다하면

printf("학생의 이름 : %s", stduent.name);

선언방법

struct 구조체이름 {

(멤버변수1자료형)(변수이름)

(멤버변수2자료형)(변수이름)

}; 마지막엔 세미콜론이 있어야한다.

**<예제>**

#include<stdio.h>

struct student {

char major[10];

char name[10];

int id;

};

int main()

{

student highschool[3];//어느 고등학교에 다니는 학생 3명에 대한 배열

highschool[0] = { "자연계", "김세영", 20210000 };

highschool[1] = { "인문계", "홍길동", 20210001 };

highschool[2] = { "자연계", "손흥민", 20210002 };

for (int i = 0; i < 3; i++) {

printf("계열 : %s 이름 : %s 학번 : %d\n", highschool[i].major, highschool[i].name, highschool[i].id);

}

}

텍스트이(가) 표시된 사진

자동 생성된 설명

추가적으로 구조체 내부에 함수도 선언할 수 있다.

#include <stdio.h>

typedef struct jun{

int x;

int y;

void co() {

printf("(%d, %d)\n", x, y);

}

}ju;

int main(){

int k;

k= 100;

ju a[100];

// 2차원 배열로 정렬했을 때보다 구조체로 묶으면 깔끔하다.

a[0].x=5; a[0].y=3;

// a라는 jun구조체 배열에 0번째 x, y값

a[0].co();

//좌표를 찍는 구조체 함수

}

문제) 5개의 좌표를 입력받고, x좌표가 오름차순이 되도록 정렬해라. 만약 x좌표가 같다면 y좌표값이 오름차순이 되도록 정렬해라.

문제) 5개의 좌표를 입력받고, x좌표가 오름차순이 되도록 정렬해라. 만약 x좌표가 같다면 y좌표값이 오름차순이 되도록 정렬해라.

#include<stdio.h>

struct point {

int x;

int y;

};

point arr[5];

bool compare(point a, point b) {

if (a.x == b.x) {

return a.y < b.y;

}

else {

return a.x < b.x;

}

}

void sort(int left, int right)//quick sort

{

if (left >= right) return;

int l = left + 1, r = right, pivot = left;

point temp;

while (l <= r)

{

for (;compare(arr[l], arr[pivot]) && l <= right; l++);

for (;compare(arr[pivot],arr[r]) && r > left; r--);

if (l < r) {

temp = arr[l];

arr[l] = arr[r];

arr[r] = temp;

l++;r--;

}

}

temp = arr[pivot];

arr[pivot] = arr[r];

arr[r] = temp;

pivot = r;

sort(left, pivot - 1); sort(pivot + 1, right);

}

int main() {

for (int i = 0, x, y; i < 5;i++) {

scanf("%d %d", &arr[i].x, &arr[i].y);

}

sort(0, 4);

printf("=============================\n");

for (int i = 0; i < 5;i++) {

printf("%d %d\n", arr[i].x, arr[i].y);

}

}

