(1-2)설명

Short는 변수에 들어가는 데이터 값의 형태가 16비트이다. 즉 정수32767까지 short자료형을 쓸 수 있다는 의미이다. I 변수는 32768의 데이터 값을 가지고 있다. 이러한 경우 선언으로부터 할당된 메모리 공간이 부족해지고 변수 i의 출력 값에 오류가 생길 수 있다. 이를 오버플로우라고 한다. 그래서 출력결과는 -32768로 잘못된 값이 나온다.

#include <stdio.h>

int main(void)

{

 short i;

 i = 32768;

 printf("i=%d\n", i);

 return 0;

}

◇ 오버플로우 개념

(1-1)설명

Short I;는 변수 선언이다. 이는 변수가 어떠한 데이터의 종류를 저장할 것인지 컴파일러에게 알려주는 과정이다. 즉 I 변수가 16비트 정수형 데이터를 저장할 수 있다는 것으로 선언이 돼있다. I는 32767의 값으로 선언이 되고 출력문에서 형식지정자를 %o,%x를 씀으로써 i가 가진 데이터를 각각 8진수와 16진수로 출력한다는 결과를 낼 수 있다.

◆ DESCRIPTION

#include <stdio.h>

int main(void)

{

 short i;

 i = 32767;

 printf("i 8진수=%o\n", i);

 printf("i 16진수=%x\n", i);

 return 0;

}

◇ 8진수,16진수 출력하기

(2-1)

자료형은 ‘변수에 들어가는 데이터 값의 형태’라는 의미이다. 이는 변수에 들어가는 데이터의 종류를 결정하는데 float가 들어가면 실수형 데이터를 저장할 수가 있다. 그래서 변수 x가 옆 소스에 적힌 데이터 값으로 초기화 되었더라도 출력할 수가 있다. 10.20f가 붙었으니 1.2345678901234567890을 추가해서 소수 20번째자리까지 출력된다.

(1-3)

Unsigned를 해석하면 부호가 없다는 의미를 지닌다. 컴퓨터는 맨 앞의 비트를 보고 음,양을 결정하는데, 예를 들어 00000001은 1이고 10000001은 -1의 값을 가진다. 하지만 unsigned는 10000001을 129로 인식해버린다. 그렇다는 것은 10000010은 130일테고 계속하면 11111111이 255의 값을 가진다. 이는 데이터 값의 형태가 약 2배로 늘어났다는 의미이니, 응용해서 보면 i변수는 32768의 데이터 값을 가질 수가 있다. (실행결과:32768)

i

#include <stdio.h>

int main(void)

{

 float x;

 x = 1.2345678901234567890;

 printf("x=%10.20f\n", x);

 return 0;

}

◇ 실수 자료형 :float

◆ DESCRIPTION

#include <stdio.h>

int main(void)

{

 unsigned short i;

 i = 32768;

 printf("i=%d\n", i);

 return 0;

}

◇ Unsigned 개념

(2-3)

할당된 메모리의 공간이 부족하면 언더플로우라는 것도 발생이 가능하다. 예를 들어 short자료형을 사용하면 -32767~32767의 값을 저장이 가능한데 -32768의 값이 저장되면 언더플로우가 발생해버린다. 옆의 소스는 실수형 자료형 float에서 언더플로우가 발생한 경우이다. 이러한 경우, 실행결과는 0으로 나오게 된다.

#include <stdio.h>

int main(void)

{

 float x;

 x = 1.23456e-46;

 printf("x=%10.20f\n", x);

 return 0;

}

◇ 실수의 언더플로우

(2-2)

1e39란 (1x10^39)라는 의미이다.

하지만 실수 float는 32비트로 위의 값을 저장할 수가 없다. 이러한 경우 할당된 데이터의 공간이 부족해서 생기는 오류가 생긴다.

1-2번 정수형의 오버플로우에서는 데이터의 값에 -가 붙는다면 실수형에서 오버플로우가 생기면 inf즉 무한대로 출력이 되버린다.

◆ DESCRIPTION

#include <stdio.h>

int main(void)

{

 float x;

 x = 1e39;

 printf("x=%10.20f\n", x);

 return 0;

}

◇ 실수의 오버플로우

(3)

형식지정자 중에서는 여러가지가 있는데 %d같은 경우 콤마 뒤에 있는 변수를 정수형 데이터로 출력하라는 의미이다. 옆 소스에서 %f는 콤마 뒤에 있는 변수를 실수형 데이터로 출력하라는 의미이다. 그리고 %e같은 경우 콤마 뒤에 있는 변수를 지수형 데이터로 출력하라는 의미이다. 실수 입력 값(x)가 89567이니 아래의 두 출력문에서는

89.567000과 8.956700e+001이 나오는 것을 알 수가 있다.

#include <stdio.h>

int main(void)

{

float x;

printf("실수를 입력하시오:");

scanf("%f", &x);

printf("실수형식으로는 %2.6f입니다.\n", x);

printf("지수형식으로는 %e입니다.\n", x);

return 0;

}

◇ scanf문과 float자료형

(2-4)

오버플로우든 언더플로우든 자료형에 맞게 할당된 메모리의 공간이 부족해서 생기는 것이니, 더 많은 비트 형태를 가지는(변수에 들어가는 데이터의 형태가 더 큰) 자료형을 쓰면 해결될 일이다. Double자료형 같은 경우, 64비트의 데이터 형태를 지닌다. 하지만 이러한 경우 float자료형을 써도 되지만 double자료형을 써서 (2-1)번과 동일한 결과를 가진다.

◆ DESCRIPTION

#include <stdio.h>

int main(void)

{

 double x;

 x = 1.2345678901234567890;

 printf("x=%10.20f\n", x);

 return 0;

}

◇ 실수 자료형: double

(5)

문제에서는 89.567을 scanf문으로 집어넣었다. 이는 x변수로 바로 저장되고 실수형으로 89.567값을 가지지만 출력문에 나오는 형식 지정자가 %e이기 때문에 콤마 뒤에 나오는 변수를 지수형태로 나타낸다.

그래서 결과는 8.956700e+01이 된다.

#include <stdio.h>

int main(void)

{

 float x;

 printf("실수를 입력하시오:");

 scanf("%f",&x);

 printf("지수 형식으로는 %e입니다.\n",x);

 return 0;

}

#include <stdio.h>

#define sangsu 3.3

int main(void)

{

float x;

 printf("평을 입력하시오:");

 scanf("%f",&x);

 printf("%2.6f평밤미터입니다.\n",x\*sangsu);

 return 0;

}

◇ 지수형식 개념

(4)

상수에 이름을 붙인 것을 기호 상수라고 한다. 기호상수의 정의 방법은 옆 소스와 같이 (#define 기호상수 이름 값)으로 정의하면 된다. 이렇게 기호 상수를 이용하면 상수에 계산해야 되는 일이 많이 발생할 때, 상수가 변결될 때 생기는 문제점을 극복할 수가 있다 옆의 소스는 30을 입력하면 상수 3.3에 scanf문으로 저장된 변수 값 30이 곱해져 99가 나옴을 알 수 있다.

◆ DESCRIPTION

◇ 기호 상수의 개념

(6)

문제에서는 scanf문을 통해서 3.9를 대입시켰다. 이 값은 실수형 데이터 형태를 가지는 변수 x에 저장된다.

이후 출력문에 나오는 (3\*x\*x\*x-7\*x\*x+9)은 3x^3-7x^2+9라는 의미이니 방정식을 c언어로 푸는 것과 마찬가지이다.

X에 저장된 데이터 값이 3.9이니 출력 값은 80.487000이 된다.

#include <stdio.h>

int main(void)

{

 float x;

 printf("x의 값을 입력하시오:");

 scanf("%f",&x);

 printf("다항식의 값은 %2.6f입니다.\n",3\*x\*x\*x-7\*x\*x+9);

 return 0;

}

◆ DESCRIPTION

◇ c언어로 방정식 풀기