\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: HW7

Author: F74086381 蘇恩質 <grgy078033@gmail.com>

Class: 112資訊系 (乙班)

Description:

規定:使用union、struct

用union和struct來包裝不同型態的資料，讓程式更readable。

由於浮點數沒辦法直接跟integer做AND計算，因此用cast和pointer的技巧來取得浮點數的address再從address中取值出來做計算。

設一個mask = 0x80000000(做float時)或是

mask = 0x8000000000000000(做double時)，也就是最高位元是1剩下都是0，與x.i或y.i做for迴圈AND運算，將x或y從最後一個bit印到第一個bit，得到floating point number的bit pattern。

sprint把argv[2]、argv[3]、argv[4]存入Sign、exp、fraction當中，並用IEEE754的算法回推原本的floating point number

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include <stdio.h>

#include <stdlib.h>

double power(int x, int y)

{

int i;

double result = 1.0;

if(y > 0)

{

for(i = 0; i < y; i++)

result \*= x;

}

else if(y < 0)

{

int z = 0 - y;

for(i = 0; i < z; i++)

result /= x;

}

return result;

}

union float\_data1 {

unsigned i;

float fd;

};

struct float\_data2 {

unsigned i;

float fd;

};

union double\_data1 {

unsigned long i;

double dd;

};

struct double\_data2 {

unsigned long i;

double dd;

};

union float\_bits1 {

struct {

char Signed;

char exp[8];

char fraction[23];

};

};

struct float\_bits2 {

char Signed;

char exp[8];

char fraction[23];

float answer;

};

union double\_bits1 {

struct {

char Signed;

char exp[11];

char fraction[52];

};

};

struct double\_bits2 {

char Signed;

char exp[11];

char fraction[52];

double answer;

};

int main(int argc, char \*argv[])

{

int z = atoi(argv[1]);

if(z == 1)

{

union float\_data1 x;

x.fd = atof(argv[2]);

x.i = \*(int \*)&x.fd;

struct float\_data2 y;

y.fd = atof(argv[2]);

y.i = \*(int \*)&y.fd;

int i;

unsigned mask = 0x80000000;

printf("Union ");

for(i = 0; i < 32; i++)

{

if(i == 0)

{

if(x.fd >= 0)

printf("0 ");

else

printf("1 ");

}

if(i >= 1 && i < 9)

{

printf("%d", ((x.i&mask)>>(31 - i)));

if(i == 8)

printf(" ");

}

if(i >= 9 && i < 32)

{

printf("%d", ((x.i&mask)>>(31 - i)));

if(i == 31)

printf("\n");

}

mask >>= 1;

}

mask = 0x80000000;

printf("Struct ");

for(i = 0; i < 32; i++)

{

if(i == 0)

{

if(x.fd >= 0)

printf("0 ");

else

printf("1 ");

}

if(i >= 1 && i < 9)

{

printf("%d", ((y.i&mask)>>(31 - i)));

if(i == 8)

printf(" ");

}

if(i >= 9 && i < 32)

{

printf("%d", ((y.i&mask)>>(31 - i)));

if(i == 31)

printf("\n");

}

mask >>= 1;

}

}

else if(z == 2)

{

union double\_data1 x;

x.dd = atof(argv[2]);

x.i = \*(long int \*)&x.dd;

struct double\_data2 y;

y.dd = atof(argv[2]);

y.i = \*(long int \*)&y.dd;

int i;

unsigned long mask = 0x8000000000000000;

printf("Union ");

for(i = 0; i < 64; i++)

{

if(i == 0)

{

if(x.dd >= 0)

printf("0 ");

else

printf("1 ");

}

if(i >= 1 && i < 12)

{

printf("%lld", ((x.i&mask)>>(63 - i)));

if(i == 11)

printf(" ");

}

if(i >= 12 && i < 64)

{

printf("%lld", ((x.i&mask)>>(63 - i)));

if(i == 63)

printf("\n");

}

mask >>= 1;

}

mask = 0x8000000000000000;

printf("Struct ");

for(i = 0; i < 64; i++)

{

if(i == 0)

{

if(y.dd >= 0)

printf("0 ");

else

printf("1 ");

}

if(i >= 1 && i < 12)

{

printf("%lld", ((y.i&mask)>>(63 - i)));

if(i == 11)

printf(" ");

}

if(i >= 12 && i < 64)

{

printf("%lld", ((y.i&mask)>>(63 - i)));

if(i == 63)

printf("\n");

}

mask >>= 1;

}

}

else if(z == 3)

{

union float\_bits1 x;

sprintf(&x.Signed, "%c", argv[2]);

sprintf(x.exp, "%8s", argv[3]);

sprintf(x.fraction, "%23s", argv[4]);

float answer\_x = 0.0;

struct float\_bits2 y;

sprintf(&y.Signed, "%c", argv[2]);

sprintf(y.exp, "%8s", argv[3]);

sprintf(y.fraction, "%23s", argv[4]);

y.answer = 0.0;

int i;

int z = atoi(argv[2]);

int binary\_exp\_x = 0;

float binary\_fra\_x = 1.0;

int binary\_exp\_y = 0;

float binary\_fra\_y = 1.0;

for(i = 7; i >= 0; i--)

binary\_exp\_x += ( power(2, 7 - i) \* (x.exp[i] - '0') );

for(i = 0; i < 23; i++)

binary\_fra\_x += ( power(2, -1 \* (i + 1) ) \* (x.fraction[i] - '0') );

answer\_x = power(2, binary\_exp\_x - 127) \* binary\_fra\_x;

if(z == 0)

{

printf("");

printf("Union %f\n", answer\_x);

}

else if(z == 1)

{

printf("Union -");

printf("%f\n", answer\_x);

}

for(i = 7; i >= 0; i--)

binary\_exp\_y += ( power(2, 7 - i) \* (y.exp[i] - '0') );

for(i = 0; i < 23; i++)

binary\_fra\_y += ( power(2, -1 \* (i + 1) ) \* (y.fraction[i] - '0') );

y.answer = power(2, binary\_exp\_y - 127) \* binary\_fra\_y;

if(z == 0)

{

printf("");

printf("Struct %f\n", y.answer);

}

else if(z == 1)

{

printf("Struct -");

printf("%f\n", y.answer);

}

}

else if(z == 4)

{

union double\_bits1 x;

sprintf(&x.Signed, "%c", argv[2]);

sprintf(x.exp, "%11s", argv[3]);

sprintf(x.fraction, "%52s", argv[4]);

float answer\_x = 0.0;

struct double\_bits2 y;

sprintf(&y.Signed, "%c", argv[2]);

sprintf(y.exp, "%11s", argv[3]);

sprintf(y.fraction, "%52s", argv[4]);

y.answer = 0.0;

int i;

int z = atoi(argv[2]);

int binary\_exp\_x = 0;

float binary\_fra\_x = 1.0;

int binary\_exp\_y = 0;

float binary\_fra\_y = 1.0;

for(i = 10; i >= 0; i--)

binary\_exp\_x += ( power(2, 10 - i) \* (x.exp[i] - '0') );

for(i = 0; i < 52; i++)

binary\_fra\_x += ( power(2, -1 \* (i + 1) ) \* (x.fraction[i] - '0') );

answer\_x = power(2, binary\_exp\_x - 1023) \* binary\_fra\_x;

if(z == 0)

{

printf("");

printf("Union %f\n", answer\_x);

}

else if(z == 1)

{

printf("Union -");

printf("%f\n", answer\_x);

}

for(i = 10; i >= 0; i--)

binary\_exp\_y += ( power(2, 10 - i) \* (y.exp[i] - '0') );

for(i = 0; i < 52; i++)

binary\_fra\_y += ( power(2, -1 \* (i + 1) ) \* (y.fraction[i] - '0') );

y.answer = power(2, binary\_exp\_y - 1023) \* binary\_fra\_y;

if(z == 0)

{

printf("");

printf("Struct %f\n", y.answer);

}

else if(z == 1)

{

printf("Struct -");

printf("%f\n", y.answer);

}

}

return 0;

}

Compilation:

gcc -o hw7 hw7.c

^

Execution:

***Example 1:***

./hw7 1 11.25

***Example 2:***

./hw7 2 11.25

***Example 3:***

./hw7 3 0 10000010 01101000000000000000000

***Example 4:***

./hw7 4 0 10000000010 0110100000000000000000000000000000000000000000000000

Output:

***Example 1:***

Union 0 10000010 01101000000000000000000

Struct 0 10000010 01101000000000000000000

***Example 2:***

Union 0 10000000010 0110100000000000000000000000000000000000000000000000

Struct 0 10000000010 0110100000000000000000000000000000000000000000000000

***Example 3:***

Union 11.250000

Struct 11.250000

***Example 4:***

Union 11.250000

Struct 11.250000