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

Report: HW7

Author: E94074029江羿賢 <e94074029@gs.ncku.edu.tw>

Class: 資訊111級乙班

Description:

在這次的作業中重新做了hw4的功能，並且利用了union跟struct來實作，而其中需要注意的是記憶體分配位置的順序，以及在裡面所需要放的資料型態為何，才能夠正確的做出相對應功能的程式，學到了很多不同的方法。

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

#include <stdio.h>

#include <stdlib.h>

void float\_to\_bit\_Struct(float);

void float\_to\_bit\_Union(float);

void double\_to\_bit\_Struct(double);

void double\_to\_bit\_Union(double);

void bit\_to\_float\_Struct(char \*, char \*, char \*);

void bit\_to\_float\_Union(char \*, char \*, char \*);

void bit\_to\_double\_Struct(char \*, char \*, char \*);

void bit\_to\_double\_Union(char \*, char \*, char \*);

unsigned long long power(int, int);

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

{

int choose = atoi(argv[1]); //choose what to use it

float a = atof(argv[2]); //the input float

double b = atof(argv[2]); //the input double

switch (choose) //choose it

{

case 1: //do float turn to bit pattern

printf("Struct ");

float\_to\_bit\_Struct(a);

printf("Union ");

float\_to\_bit\_Union(a);

break;

case 2: //do double turn to bit pattern

printf("Struct ");

double\_to\_bit\_Struct(b);

printf("Union ");

double\_to\_bit\_Union(b);

break;

case 3: //do bit pattern turn to float

printf("Struct ");

bit\_to\_float\_Struct(argv[2], argv[3], argv[4]);

printf("Union ");

bit\_to\_float\_Union(argv[2], argv[3], argv[4]);

break;

case 4: //do bit pattern turn to double

printf("Struct ");

bit\_to\_double\_Struct(argv[2], argv[3], argv[4]);

printf("Union ");

bit\_to\_double\_Union(argv[2], argv[3], argv[4]);

break;

}

}

void float\_to\_bit\_Struct(float input)

{

//use struct to do float to bit

struct

{

union {

float f;

unsigned n;

} u;

} a;

a.u.f = input; //input

int i = 32; //have 32 bits

int count = 0; //count when to print the space

while (i--)

{

if (count == 1 || count == 9)

printf(" ");

if ((a.u.n >> i) & 1) //print first bit to last bit

printf("1");

else

printf("0");

count++;

}

printf("\n");

}

void float\_to\_bit\_Union(float input)

{

//use union to do float to bit

union {

float f;

unsigned u;

} a;

a.f = input; //input

int i = 32; //have 32 bits

int count = 0; //count when to print the space

while (i--)

{

if (count == 1 || count == 9)

printf(" ");

if ((a.u >> i) & 1) //print first bit to last bit

printf("1");

else

printf("0");

count++;

}

printf("\n");

}

void double\_to\_bit\_Struct(double input)

{

//use struct to do double to bit

struct

{

union {

double d;

unsigned long long n;

} u;

} a;

a.u.d = input; //input

int i = 64; //have 64 bits

int count = 0; //count when to print the space

while (i--)

{

if (count == 1 || count == 12)

printf(" ");

if ((a.u.n >> i) & 1) //print first bit to last bit

printf("1");

else

printf("0");

count++;

}

printf("\n");

}

void double\_to\_bit\_Union(double input)

{

//use union to do double to bit

union {

double d;

unsigned long long u;

} a;

a.d = input; //input

int i = 64; //have 64 bits

int count = 0; //count when to print the space

while (i--)

{

if (count == 1 || count == 12)

printf(" ");

if ((a.u >> i) & 1) //print first bit to last bit

printf("1");

else

printf("0");

count++;

}

printf("\n");

}

void bit\_to\_float\_Struct(char \*s, char \*e, char \*m)

{

//use struct to do bit to float

struct

{

unsigned int themat : 23; //give the appropriate memory

unsigned int theexp : 8;

unsigned int thesign : 1;

} a;

unsigned int sign = 0;

unsigned int exp = 0;

unsigned int mat = 0;

int ee = 7; // exp 8 bit

int mm = 22; // mat 23 bit

if (s[0] == 48) //to see sign

sign = 0; //set to 0

else

sign = 1; //set to 1

for (int i = 0; i < 8; i++, ee--) //form 0 to 7 check exp

{

if (e[i] == 49) //count only it's 1

{

exp += power(2, ee); //to see it's exp and plus it

}

}

for (int j = 0; j < 23; j++, mm--) //form 0 to 23 check mat

{

if (m[j] == 49) //count only it's 1

{

mat += power(2, mm); //to see it's exp and plus it

}

}

a.thesign = sign;

a.theexp = exp;

a.themat = mat;

printf("%f", \*(float \*)&a); //print the float

printf("\n");

}

void bit\_to\_float\_Union(char \*s, char \*e, char \*m)

{

//use union to do bit to float

union {

float f;

struct

{

unsigned int themat : 23; //give the appropriate memory

unsigned int theexp : 8;

unsigned int thesign : 1;

} u;

} a;

unsigned int sign = 0;

unsigned int exp = 0;

unsigned int mat = 0;

int ee = 7; // exp 8 bit

int mm = 22; // mat 23 bit

if (s[0] == 48) //to see sign

sign = 0; //set to 0

else

sign = 1; //set to 1

for (int i = 0; i < 8; i++, ee--) //form 0 to 7 check exp

{

if (e[i] == 49) //count only it's 1

{

exp += power(2, ee); //to see it's exp and plus it

}

}

for (int j = 0; j < 23; j++, mm--) //form 0 to 23 check mat

{

if (m[j] == 49) //count only it's 1

{

mat += power(2, mm); //to see it's exp and plus it

}

}

a.u.themat = mat;

a.u.theexp = exp;

a.u.thesign = sign;

printf("%f", a.f); //print the union float

printf("\n");

}

void bit\_to\_double\_Struct(char \*s, char \*e, char \*m)

{

//use struct to do bit to double

struct

{

unsigned long long themat : 52; //give the appropriate memory

unsigned long long theexp : 11;

unsigned long long thesign : 1;

} a;

unsigned long long sign = 0;

unsigned long long exp = 0;

unsigned long long mat = 0;

int ee = 10; //exp 11 bit

int mm = 51; //mat 52 bit

if (s[0] == 48) //to see the sign is 0 or 1

sign = 0;

else

sign = 1;

for (int i = 0; i < 11; i++, ee--) //form 0 to 10 check exp

{

if (e[i] == 49) //count only it's 1

{

exp += power(2, ee); //to see it's exp and plus it

}

}

for (int j = 0; j < 52; j++, mm--) //form 0 to 51 to check mat

{

if (m[j] == 49) //count only it's 1

{

mat += power(2, mm); //to see it's exp and plus it

}

}

a.thesign = sign;

a.theexp = exp;

a.themat = mat;

printf("%lf", \*(double \*)&a); //print the double

printf("\n");

}

void bit\_to\_double\_Union(char \*s, char \*e, char \*m)

{

//use union to do bit to double

union {

double d;

struct

{

unsigned long long themat : 52; //give the appropriate memory

unsigned long long theexp : 11;

unsigned long long thesign : 1;

} u;

} a;

unsigned long long sign = 0;

unsigned long long exp = 0;

unsigned long long mat = 0;

int ee = 10; //exp 11 bit

int mm = 51; //mat 52 bit

if (s[0] == 48) //to see the sign is 0 or 1

sign = 0;

else

sign = 1;

for (int i = 0; i < 11; i++, ee--) //form 0 to 10 check exp

{

if (e[i] == 49) //count only it's 1

{

exp += power(2, ee); //to see it's exp and plus it

}

}

for (int j = 0; j < 52; j++, mm--) //form 0 to 51 to check mat

{

if (m[j] == 49) //count only it's 1

{

mat += power(2, mm); //to see it's exp and plus it

}

}

a.u.themat = mat;

a.u.theexp = exp;

a.u.thesign = sign;

printf("%lf", a.d); //print the union double

printf("\n");

}

unsigned long long power(int x, int y) //power function

{

unsigned long long total = 1;

while (y--) //do y times

{

total \*= x;

}

return total; //return total

}

Compilation:

gcc hw7.c -o hw7

Execution:

./hw7 1 85.125

./hw7 2 85.125

./hw7 3 0 10000101 01010100100000000000000

./hw7 4 0 10000000101 0101010010000000000000000000000000000000000000000000

Output:

Struct 0 10000101 01010100100000000000000

Union 0 10000101 01010100100000000000000

Struct 0 10000000101 0101010010000000000000000000000000000000000000000000

Union 0 10000000101 0101010010000000000000000000000000000000000000000000

Struct 85.125000

Union 85.125000

Struct 85.125000

Union 85.125000