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

Report:hw7\_1

Author: F74055047張財實 <changchaishi@gmail.com>

Class: 乙班

Description: Learned how to use typedef to optimized the call of pointer which points to function, and an array can also contains several pointers of pointer to function ,making program easy for execution.

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

#include <stdio.h>

#include <stdlib.h>

#define NUM\_FUNC 3

//For the use of function pointer array//

typedef double (\*F)(double,int);

//function prototypes

double powerpower(F,double,int,int);

double power(double, int);

double multiply(double, int);

double divide(double, int);

//main function,i can say no more//

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

F func\_ptr[NUM\_FUNC]={power,multiply,divide};

double x = atof(argv[1]);

int n = atoi(argv[2]);

int m = atoi(argv[3]);

int j;

for(j=0;j<NUM\_FUNC;j++){

printf("%f\n",powerpower(func\_ptr[j],x,n,m));

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

three functions which have to be pointed by type F

F is a pointer to funtion

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double power(double x,int n){

int i;

double value=1.0f;

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

value\*=x;

}

return value;

}

double multiply(double x,int n){

return (double)(x\*n);

}

double divide(double x,int n){

return (double)(x/n);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fucntion that calles pointers to pass the power

multiply and divide function to itself for the calculation

of powerpower

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double powerpower(F func\_ptr,double x,int n,int m){

double value=1;

int i;

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

value\*=(func\_ptr(x,n));

}

return value;

}

Compilation:

gcc –o hw7\_1 hw7\_1.c

Execution:

./hw7\_1 3 2 3

Output:

729.000000

216.000000

3.375000

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

Report: hw7\_2

Author: F74055047張財實 <changchaishi@gmail.com>

Class: 乙班

Description:

Union is actually a replacement of the need of pointer to make the float as an integer type. So, union’s members share the same memory location which bit pattern will be same. Assigning a float to union, we can just simply take the integer for the bitwise operation.

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

#include <stdio.h>

#include <stdlib.h>

union binary{

int int\_x;

float float\_x;

};

//function prototype

void bit\_pattern(union binary u);

//main :)

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

union binary u;

int N;

int i;

N=atoi(argv[1]);

switch(N){

case 1:

u.int\_x=atoi(argv[2]);

bit\_pattern(u);

break;

case 2:

u.float\_x=atof(argv[2]);

bit\_pattern(u);

break;

case 3:

//setting the bit integer type

u.int\_x=0;

for(i=0;i<31;i++){

if((argv[2][i]-'0')==1)

u.int\_x|=1<<(31-i);

}

printf("%.23e\n",u.float\_x);

break;

}

return 0;

}

//prints the bit pattern by using bitwise operation

void bit\_pattern(union binary u){

int i;

for(i=31;i>=0;i--){

if(u.int\_x&1<<i)

printf("1");

else

printf("0");

}

printf("\n");

}

Compilation:

gcc –o hw7\_2 hw7\_2.c

Execution:

1)./hw7\_2 1 -1

2)./hw7\_2 2 2

3)./hw7\_2 3

Output:

1)11111111111111111111111111111111

2)01000000000000000000000000000000

3)1.17549771393860188752971e-38

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

Report: hw7\_3

Author: F74055047張財實 <changchaishi@gmail.com>

Class: 乙班

Description:

Using bit-field is just a replacement of bitwise operation. With the characteristic of union, we declare 32 variable with 1 bit-field in a structure inside the union.it cuts the 32 bit value (float or int) in 32 parts, by another union differs two types of data input. An huge union can let us print bit, set bit without point and bitwise operators.

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

#include <stdio.h>

#include <stdlib.h>

void print\_bit(void);

union{

struct{

//cute 32 parts

unsigned int b1:1;unsigned int b2:1;

unsigned int b3:1;unsigned int b4:1;

unsigned int b5:1;unsigned int b6:1;

unsigned int b7:1;unsigned int b8:1;

unsigned int b9:1;unsigned int b10:1;

unsigned int b11:1;unsigned int b12:1;

unsigned int b13:1;unsigned int b14:1;

unsigned int b15:1;unsigned int b16:1;

unsigned int b17:1;unsigned int b18:1;

unsigned int b19:1;unsigned int b20:1;

unsigned int b21:1;unsigned int b22:1;

unsigned int b23:1;unsigned int b24:1;

unsigned int b25:1;unsigned int b26:1;

unsigned int b27:1;unsigned int b28:1;

unsigned int b29:1;unsigned int b30:1;

unsigned int b31:1;unsigned int b32:1;

}s;

union{

//two types

int int\_x;

float float\_x;

}u;

}type\_u;

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

int \*A;

int N;

int i;

N=atoi(argv[1]);

switch(N){

case 1:

type\_u.u.int\_x=atoi(argv[2]);

print\_bit();

break;

case 2:

type\_u.u.float\_x=atof(argv[2]);

print\_bit();

break;

case 3:

//assign each "bit" in each part of memory location

type\_u.s.b1=argv[2][31]-'0';

type\_u.s.b2=argv[2][30]-'0';

type\_u.s.b3=argv[2][29]-'0';

type\_u.s.b4=argv[2][28]-'0';

type\_u.s.b5=argv[2][27]-'0';

type\_u.s.b6=argv[2][26]-'0';

type\_u.s.b7=argv[2][25]-'0';

type\_u.s.b8=argv[2][24]-'0';

type\_u.s.b9=argv[2][23]-'0';

type\_u.s.b10=argv[2][22]-'0';

type\_u.s.b11=argv[2][21]-'0';

type\_u.s.b12=argv[2][20]-'0';

type\_u.s.b13=argv[2][19]-'0';

type\_u.s.b14=argv[2][18]-'0';

type\_u.s.b15=argv[2][17]-'0';

type\_u.s.b16=argv[2][16]-'0';

type\_u.s.b17=argv[2][15]-'0';

type\_u.s.b18=argv[2][14]-'0';

type\_u.s.b19=argv[2][13]-'0';

type\_u.s.b20=argv[2][12]-'0';

type\_u.s.b21=argv[2][11]-'0';

type\_u.s.b22=argv[2][10]-'0';

type\_u.s.b23=argv[2][9]-'0';

type\_u.s.b24=argv[2][8]-'0';

type\_u.s.b25=argv[2][7]-'0';

type\_u.s.b26=argv[2][6]-'0';

type\_u.s.b27=argv[2][5]-'0';

type\_u.s.b28=argv[2][4]-'0';

type\_u.s.b29=argv[2][3]-'0';

type\_u.s.b30=argv[2][2]-'0';

type\_u.s.b31=argv[2][1]-'0';

type\_u.s.b32=argv[2][0]-'0';

printf("%.23e\n",type\_u.u.float\_x);

break;

}

return 0;

}

//function to print each bit-field's "bit"

void print\_bit(void){

printf("%d",type\_u.s.b32);printf("%d",type\_u.s.b31);

printf("%d",type\_u.s.b30);printf("%d",type\_u.s.b29);

printf("%d",type\_u.s.b28);printf("%d",type\_u.s.b27);

printf("%d",type\_u.s.b26);printf("%d",type\_u.s.b25);

printf("%d",type\_u.s.b24);printf("%d",type\_u.s.b23);

printf("%d",type\_u.s.b22);printf("%d",type\_u.s.b21);

printf("%d",type\_u.s.b20);printf("%d",type\_u.s.b19);

printf("%d",type\_u.s.b18);printf("%d",type\_u.s.b17);

printf("%d",type\_u.s.b16);printf("%d",type\_u.s.b15);

printf("%d",type\_u.s.b14);printf("%d",type\_u.s.b13);

printf("%d",type\_u.s.b12);printf("%d",type\_u.s.b11);

printf("%d",type\_u.s.b10);printf("%d",type\_u.s.b9);

printf("%d",type\_u.s.b8);printf("%d",type\_u.s.b7);

printf("%d",type\_u.s.b6);printf("%d",type\_u.s.b5);

printf("%d",type\_u.s.b4);printf("%d",type\_u.s.b3);

printf("%d",type\_u.s.b2);printf("%d\n",type\_u.s.b1);

}

Execution:

1)./hw7\_2 1 -1

2)./hw7\_2 2 2

3)./hw7\_2 3

Output:

1)11111111111111111111111111111111

2)01000000000000000000000000000000

3)1.17549771393860188752971e-38