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

Report: HW3

Author: F74071027 學貝萁 <youqing1211@gmail.com>

Class: 甲班

Description:

　　這個程式主要在說明，一個資料會以16進位存在記憶體中，而不同的資料型態會決定這個資料的呈現方式，也就是在不同的資料型態中做轉換（我應該沒說錯吧）。

　　讀入以二進位表示的數時因為每個單一數字都要讀到，所以配合陣列與ASCII將此數轉進陣列（有試過直接使用atoi轉換但編譯會出現錯誤，未查明原因），至於整數與浮點數則使用atoi與atof即可。

　　轉換資料型態時，則先將二進位轉換為整數，再使用指標將整數轉換為浮點數；反之亦然，就算是小數轉二進位，也先使用指標將其轉為整數後再轉為二進位。程式碼中，如果為負數的話，我選擇在整數型態時扣掉231或263，如此一來可以減少在迴圈內計算的時間。

　　另外要注意的一點是，在輸出long long int或double時，必須將%d或%f改成%lld或%lf，否則依然會以int 或float的形式呈現（我在這裡卡了有點久，去翻教授簡報還有上網查才發現這件事…）。

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

#include<stdio.h>

#include<stdlib.h>

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

int M=atoi(argv[1]),i,j;

if(M==1){ //32-bit > int & float

int Q[32],A=0,n=1;//n:positive=0 or negative=1

for(i=0;i<32;i++)Q[i]=argv[2][i]-48;

//===int===

if(Q[0]==0)n=0; //positive

for(i=n;i<32;i++)if(Q[i]==1){

for(j=0;j<31-i;j++)Q[i]=Q[i]\*2;

A=A+Q[i];

}

if(n==1)A=A-2147483648;

printf("integer: %d\n",A);

//===float===

float \*pA;

pA=&A;

printf("float: %f\n",\*pA);

}

///////////////////////////////////////////

else if(M==2){ //int > 32-bit

int Q=atoi(argv[2]),A[32]={0},arrow=31,n=0;//positive

if(Q<0){

Q=Q+2147483648;

n=1;//negative

}

while(Q!=0){

A[arrow]=Q%2;

Q=Q/2;

arrow--;

}

if(n==0)printf("32-bit: ");

else printf("32-bit: 1");

for(i=n;i<32;i++)printf("%d",A[i]);

printf("\n");

}

///////////////////////////////////////////

else if(M==3){ //float > 32-bit

float Q=atof(argv[2]);

int \*pQ;

pQ=&Q;

int Q1=\*pQ;

//======

int A[32]={0},arrow=31,n=0;//positive

if(Q1<0){

Q1=Q1+2147483648;

n=1;//negative

}

while(Q1!=0){

A[arrow]=Q1%2;

Q1=Q1/2;

arrow--;

}

if(n==0)printf("32-bit: ");

else printf("32-bit: 1");

for(i=n;i<32;i++)printf("%d",A[i]);

printf("\n");

}

///////////////////////////////////////////

else if(M==4){ //64-bit > int & double

long long int Q[64],A=0,n=1;//negative

for(i=0;i<64;i++)Q[i]=argv[2][i]-48;

//===int===

if(Q[0]==0)n=0; //positive

for(i=n;i<64;i++)if(Q[i]==1){

for(j=0;j<63-i;j++)Q[i]=Q[i]\*2;

A=A+Q[i];

}

if(n==1)A=A-9223372036854775808;

printf("integer: %lld\n",A);

//===double===

double \*pA;

pA=&A;

printf("double: %lf\n",\*pA);

}

///////////////////////////////////////////

else if(M==5){ //int > 64-bit

long long int Q=atoll(argv[2]),A[64]={0},arrow=63,n=0;//positive

if(Q<0){

Q=Q+9223372036854775808;

n=1;//negative

}

while(Q!=0){

A[arrow]=Q%2;

Q=Q/2;

arrow--;

}

if(n==0)printf("64-bit: ");

else printf("64-bit: 1");

for(i=n;i<64;i++)printf("%lld",A[i]);

printf("\n");

}

///////////////////////////////////////////

else if(M==6){ //double > 64-bit

double Q=atof(argv[2]);

long long int \*pQ;

pQ=&Q;

long long int Q1=\*pQ;

//======

long long int A[64]={0},arrow=63,n=0;//positive

if(Q1<0){

Q1=Q1+9223372036854775808;

n=1;//negative

}

while(Q1!=0){

A[arrow]=Q1%2;

Q1=Q1/2;

arrow--;

}

if(n==0)printf("64-bit: ");

else printf("64-bit: 1");

for(i=n;i<64;i++)printf("%lld",A[i]);

printf("\n");

}

return 0;

}

Compilation:

gcc hw3.c -o hw3

Execution:

./hw3 1 11100000000000000000000000000000

integer: -536870912

float: -36893488147419103232.000000

./hw3 2 111

32-bit: 00000000000000000000000001101111

./hw3 3 111

32-bit: 01000010110111100000000000000000

./hw3 4 0100000000101001100000000000000000000000000000000000000000000000

integer: 4623367229960880128

double: 12.750000

./hw3 5 4623367229960880128

64-bit: 0100000000101001100000000000000000000000000000000000000000000000

./hw3 6 12.75

64-bit: 0100000000101001100000000000000000000000000000000000000000000000