|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
|  | | |
| Практическое задание № 2 | | |
| по дисциплине «Информатика» | | |
|  | | |
| **ПРЕДСТАВЛЕНИЕ ДАННЫХ В ЭВМ** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-81 |
| Вариант: | 9 |
| Студенты: | Демидович Егор |
|  | Муравьев Михаил |
|  |  |
| Преподаватели: | Тимофеева Анастасия Юрьевна |
|  |  |
|  | | |
| Новосибирск | | |
| 2019 | | |

1. **Условие задачи**

Разработать программы для следующих задач:

- Интерпретируя действительное число как число с плавающей запятой, записать его шестнадцатеричное восьмибайтное и четырехбайтное представление.

-Преобразовать шестнадцатеричный восьмибайтный и четырехбайтный код в число с плавающей запятой.

1. **Анализ задачи**

Для первой задачи: число переводится в двоичную систему и делится на группы по 4 цифры (тетрады). Знак, порядок и мантисса переводятся отдельно друг от друга, а потом объединяются в одно число в двоичной системе, далее каждая тетрада переводится в соответствующий символ 16-ричной системы счисления. Данные операции выполняются для двух типов данных float(четырехбайтное представление) и double(восьмибайтное).

Для второй задачи: Заданное число в 16-ричном представлении переводится в вещественное число.

1. **Текст программы**

#include <stdio.h>

#include <conio.h>

#include <string>

#include <iostream>

#include <fstream>

using namespace std;

const char k\_base\_symbols[] = "0123456789ABCDEFGHIJKLMNOPQRASUVWXYZ";

char inttochar(int a) {

return k\_base\_symbols[a];

}

int chartoint(char a) {

int i = 0;

while (k\_base\_symbols[i] != a)

i++;

return i;

}

double fromRto10(string num, int r) {

double ans = 0;

int c = 0;

int end = 0;

while (num[c] != '\0' && num[c] != '.')

c++;

end = c;

while (num[end] != '\0')

end++;

for (int i = 0; i < c; i++)

ans += chartoint(num[c - 1 - i]) \* powf(r, i);

for (int i = -1; i > c - end; i--)

ans += chartoint(num[c - i]) \* powf(r, i);

return ans;

}

string from10toR(double num, int r) {

if (num == 0)

return "0";

char ans[220];

int int\_num = num;

double frac\_num = num - floor(num);

int shift = 108;

int i = shift;

int j;

int iterator = 0;

while (int\_num - powf(r, i) < 0)

i--;

while (i >= 0)

{

j = r - 1;

while (j >= 0 && int\_num - j \* powf(r, i) < 0)

j--;

int\_num -= j \* powf(r, i);

ans[iterator++] = inttochar(j);

i--;

}

if (frac\_num == 0) {

ans[iterator] = '\0';

return ans;

}

if (iterator == 0)

ans[iterator++] = '0';

ans[iterator++] = '.';

i = -1;

while (i >= -shift)

{

j = r - 1;

while (j >= 0 && frac\_num - j \* powf(r, i) < 0)

j--;

frac\_num -= j \* powf(r, i);

ans[iterator++] = inttochar(j);

i--;

}

while (ans[iterator-1] == '0')

iterator--;

ans[iterator] = '\0';

return ans;

}

string bintohex(string bin) {

string hex;

float num = fromRto10(bin, 2);

hex = from10toR(num, 16);

return hex;

}

string hextobin(string hex) {

string bin;

float dec = fromRto10(hex, 16);

int c = 8;

while (c != 0)

{

bin.append((dec >= c) ? "1" : "0");

if (dec >= c)

dec -= c;

c /= 2;

}

return bin;

}

string getbinorder(double num, int bites) {

int power = 0;

int i = 0;

int dota = 0;

int end = 0;

string m = from10toR(abs(num), 2);

while (m[i] != '\0' && m[i] != '.')

i++;

dota = i;

while (m[i] != '\0')

i++;

end = i;

power = dota - 1;

float order = power + (pow(2, bites - 1) - 1);

string binorder = from10toR(order, 2);

return binorder;

}

string getbinmantis(double num, int bites) {

string m = from10toR(abs(num), 2);

char s[100];

int i = 1;

int iterator = 0;

for (int j = 0; j < bites; j++)

{

s[j] = '0';

}

s[bites] = '\0';

while (m[i] != '\0')

{

if (m[i] != '.')

s[iterator++] = m[i];

i++;

}

return s;

}

string getbinfloat(double num) {

int powerbites = 8;

int mantissabites = 23;

string binfloat;

bool signb = num < 0;

string sign = signb ? "1" : "0";

string binorder = getbinorder(num, powerbites);

string binmantis = getbinmantis(num, mantissabites);

binfloat.append(sign);

binfloat.append(binorder);

binfloat.append(binmantis);

return binfloat;

}

string getbindouble(double num) {

int powerbites = 11;

int mantissabites = 52;

string bindouble;

bool signb = num < 0;

string sign = signb ? "1" : "0";

string binorder = getbinorder(num, powerbites);

string binmantis = getbinmantis(num, mantissabites);

bindouble.append(sign);

bindouble.append(binorder);

bindouble.append(binmantis);

return bindouble;

}

string gethexfloat(double num) {

string hexfloat;

string binfloat = getbinfloat(num);

for (int i = 0; i < 32; i = i+4)

{

float p1 = fromRto10(binfloat.substr(i, 4),2);

string p2 = from10toR(p1, 16);

hexfloat.append(p2);

}

return hexfloat;

}

string gethexdouble(double num) {

string hexdouble;

string bindouble = getbindouble(num);

for (int i = 0; i < 64; i = i + 4)

{

hexdouble.append(bintohex(bindouble.substr(i, 4)));

}

return hexdouble;

}

string getbinhex(string hex) {

string bin;

int i = 0;

while (hex[i] != '\0')

{

bin.append(hextobin(hex.substr(i,1)));

i++;

}

return bin;

}

float getfloatbin(string bins) {

int powerbites = 8;

int mantissabites = 23;

bool sign;

string order;

string mantis = "1.";

int power;

float ans = 0;

sign = bins[0] == '1';

order = bins.substr(1, powerbites);

mantis.append(bins.substr(powerbites + 1, mantissabites));

power = fromRto10(order, 2) - pow(2, powerbites - 1) + 1;

ans = fromRto10(mantis, 2) \* pow(2, power) \* (sign == 1?-1:1);

return ans;

}

double getdoublebin(string bins) {

int powerbites = 11;

int mantissabites = 52;

bool sign;

string order;

string mantis = "1.";

int power;

double ans = 0;

sign = bins[0] == '1';

order = bins.substr(1, powerbites);

mantis.append(bins.substr(powerbites + 1, mantissabites));

power = fromRto10(order, 2) - pow(2, powerbites - 1) + 1;

ans = fromRto10(mantis, 2) \* pow(2, power) \* (sign == 1 ? -1 : 1);

return ans;

}

double getallhex(string hex) {

string bins = getbinhex(hex);

double num = 0;

if (bins.size() == 32)

num = getfloatbin(bins);

else if (bins.size() == 64)

num = getdoublebin(bins);

return num;

}

void v1() {

ifstream f;

f.open("input1.txt");

if ((!f.is\_open()) || (f.peek() == EOF)) return;

double num;

while (!f.eof())

{

f >> num;

string numf = gethexfloat(num);

string numd = gethexdouble(num);

printf\_s("num:\t\t%f\n", num);

printf("numfloat:\t %s\n", numf.c\_str());

printf("numdouble:\t %s\n\n", numd.c\_str());

}

}

void v2() {

ifstream f;

f.open("input2.txt");

if ((!f.is\_open()) || (f.peek() == EOF)) return;

string hex;

double num;

while (!f.eof())

{

f >> hex;

num = getallhex(hex);

printf\_s("hex:\t\t%s\n", hex.c\_str());

printf\_s("num:\t\t%f\n\n", num);

}

}

int main() {

v1();

printf\_s("=====================================\n\n");

v2();

}

1. **Тесты**

Подпрограмма 1:

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод float | Вывод double |
| -75.5625 | C2972000 | C052E40000000000 |
| 483.5 | 43F1C000 | 407E380000000000 |
| -33.0781 | C2044FF9 | C04089FF2E48E8A7 |
| 223.352 | 435F5A1C | 406BEB4395810625 |

Подпрограмма 2:

|  |  |
| --- | --- |
| Ввод | Вывод |
| C3F8C800 | -497.562500 |
| 438E5000 | 284.625000 |
| C07D83E000000000 | -472.242188 |
| 403F400000000000 | 31.250000 |