

Федеральное государственное бюджетное образовательное учреждение
высшего образования Нижегородский государственный технический
университет им. Р.Е. Алексеева

Лабораторная работа №1
«Представление вещественных чисел памяти ЭВМ»

Выполнил:
студент 2-го курса
кафедры «Прикладная математика»
Поляков Кирилл

2020г.

Оглавление

Введение.....	3
Документация.....	4
Файл CBN.cpp.....	4
Содержание CBN.cpp.....	5
Файл Conversion.h.....	6
Класс Conversion.....	7
Содержание Conversion.h.....	12
Содержание Conversion.cpp.....	15
Тестирование.....	17

Введение

Цель работы - написать программу, принимающую на вход вещественное число и переводящую его в двоичное представление. Для этого нужно будет изучить два типа преобразования вещественного числа: классический (с помощью вычисления экспоненты и мантиссы) и с помощью Union объединения.

CBN.cpp File Reference

```
#include <iostream>
#include "Conversion.h"
```

Functions

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

Function Documentation

◆ **main()**

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

Входная точка программы

Parameters

argc Количество аргументов командной строки

argv[] Аргументы командной строки

Returns

int

Definition at line **13** of file **CBN.cpp**.

CBN.cpp

```
1 #include <iostream>
2 #include "Conversion.h"
3
4 using namespace std;
5
13 int main(int argc, char* argv[])
14 {
15     cout << Conversion::ConvertToBinary(argc, argv) << endl;
16
17     getchar();
18 }
19
```

Conversion.h File Reference

```
#include <string>
#include <iostream>
#include <math.h>
#include <Windows.h>
```

Classes

class **Conversion**

Enumerations

enum **types** { **Float** = 0, **Double** }

Conversion Class Reference

```
#include <Conversion.h>
```

Static Public Member Functions

```
template<class T >  
static string ConvertToBinary (T value, string typeConvert, string typeValue)  
static string ConvertToBinary (int argc, char *argv[])
```

Static Private Member Functions

```
template<class T >  
static string IntPart (T value)  
template<class T >  
static string FractionalPart (T value)  
static string CalculationExponent (string intPart, string fractionalPart, int type)  
static string CalculationMantissa (string intPart, string fractionalPart, int type)  
template<class T >  
static string StandartConvert (T value)  
template<class T >  
static string UnionConvert (T value)  
static void EditWindow (string typeValue, string typeConvert, double value)
```

Detailed Description

Класс для преобразования числа в бинарный код

Definition at line **13** of file **Conversion.h**.

Member Function Documentation

◆ **CalculationExponent()**

```
string Conversion::CalculationExponent ( string intPart,
                                         string fractionalPart,
                                         int    type
                                         )
```

static private

Функция вычисления экспоненты

Parameters

intPart Целочисленная часть исходного числа
fractionalPart Дробная часть исходного числа
type Тип числа (float/double)

Returns

Строка, содержащая экспоненту

Definition at line 5 of file [Conversion.cpp](#).

◆ CalculationMantissa()

```
string Conversion::CalculationMantissa ( string intPart,
                                         string fractionalPart,
                                         int    type
                                         )
```

static private

Функция вычисления мантиссы

Parameters

intPart Целочисленная часть исходного числа
fractionalPart Дробная часть исходного числа
type Тип числа (float/double)

Returns

Строка, содержащая мантиссу

Definition at line 32 of file [Conversion.cpp](#).

◆ ConvertToBinary() [1/2]


```
string Conversion::ConvertToBinary ( int    argc,
                                     char * argv[]
                                     )
```

static

Функция преобразования числа в бинарный код

Parameters

argc Количество аргументов командной строки

argv[] Аргументы командной строки

Returns

Исходное число в бинарной записи

Definition at line **95** of file **Conversion.cpp**.

◆ ConvertToBinary() [2/2]

```
template<class T >
```

```
string Conversion::ConvertToBinary ( T    value,
                                     string typeConvert,
                                     string typeValue
                                     )
```

static

Функция преобразования числа в бинарный код

Parameters

value Исходное число

typeConvert Тип преобразования

typeValue Тип исходного числа(float/double)

Returns

Исходное число в бинарной записи

Definition at line **101** of file **Conversion.h**.

◆ EditWindow()

```
void Conversion::EditWindow ( string  typeValue,  
                             string  typeConvert,  
                             double  value  
                             )
```

static private

Функция для изменения окна командной строки

Parameters

typeValue Тип исходного числа (float/double)
typeConvert Тип преобразования числа (Union/Classic)
value Исходное число

Definition at line 60 of file [Conversion.cpp](#).

◆ FractionalPart()

```
template<class T >
```

```
string Conversion::FractionalPart ( T  value )
```

static private

Функция выделения дробной части и преобразование её в бинарный код

Parameters

value Переданное значение

Returns

Строка, содержащая двоичное представление дробной части числа

Definition at line 216 of file [Conversion.h](#).

◆ IntPart()

```
template<class T >
```

```
string Conversion::IntPart ( T  value )
```

static private

Функция выделения целочисленной части и преобразование её в бинарный код

Parameters

value Переданное значение

Returns

Строка, содержащая двоичное представление целой части числа

Definition at line 192 of file [Conversion.h](#).

◆ StandartConvert()

```
template<class T >
```

```
string Conversion::StandartConvert ( T value )
```

staticprivate

Функция для стандартного(экспонента + мантисса) преобразования числа

Parameters

value Исходное число

Returns

Исходное число в бинарной записи

Definition at line **125** of file **Conversion.h**.

◆ UnionConvert()

```
template<class T >
```

```
string Conversion::UnionConvert ( T value )
```

staticprivate

Функция для преобразования числа с помощью объединения Union

Parameters

value Исходное число

Returns

Исходное число в бинарной записи

Definition at line **160** of file **Conversion.h**.

The documentation for this class was generated from the following files:

- **Conversion.h**
- **Conversion.cpp**

Conversion.h

```
1  #pragma once
2
3  #include <string>
4  #include <iostream>
5  #include <math.h>
6  #include <Windows.h>
7
8  using namespace std;
9
13 class Conversion
14 {
15     template <class T>
22     static string IntPart(T value);
23
24     template <class T>
31     static string FractionalPart(T value);
32
41     static string CalculationExponent(string intPart, string fractionalPart, int type);
50     static string CalculationMantissa(string intPart, string fractionalPart, int type);
51
52     template <class T>
59     static string StandartConvert(T value);
60     template <class T>
67     static string UnionConvert(T value);
68
76     static void EditWindow(string typeValue, string typeConvert, double value);
77
78 public:
79     template <class T>
88     static string ConvertToBinary(T value, string typeConvert, string typeValue);
96     static string ConvertToBinary(int argc, char* argv[]);
97 };
98
99 //-----//
100 template <class T>
101 string Conversion::ConvertToBinary(T value, string typeConvert, string typeValue)
102 {
103     if ((typeValue != "double") && (typeValue != "float"))
104         return "Error. Unknown value type.";
105
106     if (typeConvert == "classic")
107     {
108         if (typeValue == "float")
109             return StandartConvert((float)value);
110         else if (typeValue == "double")
111             return StandartConvert((double)value);
112     }
113     else if (typeConvert == "union")
114     {
115         if (typeValue == "float")
116             return UnionConvert((float)value);
117         else if (typeValue == "double")
118             return UnionConvert((double)value);
119     }
120     else
121         return "Error. Unknown conversion type.";
122 }
123
124 template <class T>
125 string Conversion::StandartConvert(T value)
126 {
127
128     EditWindow(string(typeid(T).name()), "classic", (double)value);
129
130
131     string result;
132
133     if (value > 0)
134         result += "0";
135     else
136         result += "1";
137 }
```

```

138     int type = 0;
139     if (string(typeid(value).name()) == "double")
140         type = 1;
141
142     result += CalculationExponent(IntPart(value), FractionalPart(value), type);
143     result += CalculationMantissa(IntPart(value), FractionalPart(value), type);
144
145     int temp = 0;
146     for (int i = 0; i < result.length(); i++)
147     {
148         temp++;
149         if (temp == 9)
150         {
151             result.insert(i, " ");
152             temp = 0;
153         }
154     }
155
156     return result;
157 }
158
159 template <class T>
160 string Conversion::UnionConvert(T value)
161 {
162     EditWindow(string(typeid(T).name()), "union", (double)value);
163
164
165     union
166     {
167         T value;
168         char bytes[sizeof(T)];
169     }temp;
170
171     bool fl = false;
172
173     temp.value = value;
174     string str;
175
176     if ((string(typeid(value).name()) == "float") || (string(typeid(value).name()) == "do
177     {
178         for (int i = sizeof(T) - 1; i >= 0; i--)
179         {
180             for (int j = 7; j >= 0; j--)
181                 str += to_string(((temp.bytes[i] >> j) & 1));
182             str += " ";
183         }
184         return str;
185     }
186     return "Error";
187 }
188
189 enum types { Float = 0, Double };
190
191 template <class T>
192 string Conversion::IntPart(T value)
193 {
194     int intPart = floor(abs(value));
195     string result;
196
197     if (intPart != 0)
198         while (intPart > 0)
199         {
200             int temp = intPart % 2;
201             if (temp == -1)
202                 temp = 1;
203             result += (temp + '0');
204             intPart /= 2;
205         }
206     else
207         result = "0";
208
209     reverse(result.begin(), result.end());
210
211     return result;
212 }
213
214
215 template <class T>
216 string Conversion::FractionalPart(T value)
217 {

```

```

218     string result;
219     double intPart, fractionalPart;
220
221     fractionalPart = modf(abs(value), &intPart);
222
223     if (fractionalPart != 0)
224     {
225         int temp = 23;
226         if (string(typeid(value).name()) == "double")
227             temp = 52;
228         for (;;)
229         {
230             while (fractionalPart != 1 && result.length() < temp)
231             {
232                 fractionalPart *= 2;
233                 result += (int)(fractionalPart) + '0';
234                 if (fractionalPart > 1)
235                     fractionalPart = modf(fractionalPart, &intPart);
236             }
237             return result;
238         }
239     }
240     else
241     {
242         result = "0";
243         return result;
244     }
245 }
246 }
247

```

Conversion.cpp

```
1 #include "Conversion.h"
2 #include <algorithm>
3 #include <iomanip>
4
5 string Conversion::CalculationExponent(string intPart, string fractionalPart, int type)
6 {
7     string exponent;
8
9     if (intPart.size() > 0)
10    {
11        int exp = intPart.size() - 1;
12        if (type == Float)
13            exp += 127;
14        else if (type == Double)
15            exp += 1023;
16
17        while (exp > 0)
18        {
19            exponent += (exp % 2 + '0');
20            exp /= 2;
21        }
22
23        reverse(exponent.begin(), exponent.end());
24        return exponent;
25    }
26    else
27        CalculationExponent(fractionalPart, intPart, type);
28
29 }
30
31
32 string Conversion::CalculationMantissa(string intPart, string fractionalPart, int type)
33 {
34     string mantissa;
35
36     if (intPart.size() > 0)
37     {
38         mantissa += intPart.erase(0, 1);
39         mantissa += fractionalPart;
40     }
41     else
42     {
43         mantissa += intPart;
44         mantissa += fractionalPart.erase(0, 1);
45     }
46     if (type == Float || type == Double)
47     {
48         int beg = 23;
49         if (type == Double)
50             beg = 52;
51         int temp = beg - mantissa.size();
52         for (unsigned int i = 0; i < temp; i++)
53             mantissa += '0';
54     }
55
56     return mantissa;
57 }
58
59
60 void Conversion::EditWindow(string typeValue, string typeConvert, double value)
61 {
62     wchar_t szTITLE[] = L"CONVERTER";
63
64     int temp = 1;
65     int _temp = 0;
66     if (typeValue == "double")
67     {
68         temp = 2;
69         _temp = 45;
70     }
71
72     SetConsoleTitle(szTITLE);
73     Sleep(1.5);
```

```

74 MoveWindow(FindWindow(NULL, szTITLE), 1, 1, 440 * temp - _temp, 220, false);
75
76 for (int i = 0; i < 36 * temp; i++)
77     cout << "-";
78 cout << endl;
79 system("cls");
80
81 string _typeValue, _typeConvert;
82
83 transform(typeValue.begin(), typeValue.end(), typeValue.begin(), toupper);
84 transform(typeConvert.begin(), typeConvert.end(), typeConvert.begin(), toupper);
85
86 cout << typeConvert << " conversion of a number" << endl;
87 cout << "'" << value << "'" << endl;
88 cout << "from " << typeValue << " to binary." << endl;
89
90 for (int i = 0; i < 36 * temp; i++)
91     cout << "-";
92 cout << endl;
93 }
94
95 string Conversion::ConvertToBinary(int argc, char* argv[])
96 {
97     string value, typeValue, typeConvert;
98
99     for (int i = 0; i < argc; i++)
100     {
101         string temp = argv[i];
102         if (temp.find("--number=") != string::npos)
103             value = temp.substr(temp.find('=') + 1, temp.length() - 1);
104         else if (temp.find("--number") != string::npos && ++i < argc)
105             value = string(argv[i]);
106
107         if (temp.find("--method=") != string::npos)
108             typeConvert = temp.substr(temp.find('=') + 1, temp.length() - 1);
109         else if (temp.find("--method") != string::npos && ++i < argc)
110             typeConvert = string(argv[i]);
111
112         if (temp.find("--type=") != string::npos)
113             typeValue = temp.substr(temp.find('=') + 1, temp.length() - 1);
114         else if (temp.find("--type") != string::npos && ++i < argc)
115             typeValue = string(argv[i]);
116     }
117
118     if ((typeValue != "double") && (typeValue != "float"))
119     {
120         cout << "Type Value:" << typeValue << "!" << endl;
121         return "Error. Unknown value type.";
122     }
123     if ((typeConvert != "union") && (typeConvert != "classic"))
124         return "Error. Unknown conversion type.";
125
126
127
128     if (typeValue == "float")
129         return ConvertToBinary(stof(value), typeConvert, typeValue);
130     else if (typeValue == "double")
131         return ConvertToBinary(stod(value), typeConvert, typeValue);
132
133
134 }

```


Тестирование

```
CONVERTER
'-1.25'
from DOUBLE to binary.
-----
11111111 11101000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
CONVERTER
UNION conversion of a number
'-1.25'
from FLOAT to binary.
-----
10111111 10100000 00000000 00000000
```