Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ПОЗИЦИОННЫЕ СИСТЕМЫ СЧИСЛЕНИЯ

Лабораторная работа № 2
по курсу «Структуры и алгоритмы обработки данных»

Вариант № 4

Выполнил: студент группы 220601 _________ Белым А.А.
(подпись)
Проверил: д. ф.-м.н, проф.каф. АТМ _________ Двоенко С.Д.
(подпись)

Тула 2013

Цель работы

Ознакомиться с позиционными системами счисления и их применением в ЭВМ. Написать программу работы с числами в разных системах счисления.

Задание

Напишите программу перехода от записи числа в обратном двоичном коде $\pm\left(a_n \ldots a_0\right)_2$ к его нега-двоичной записи $\left(b_{n+1} \ldots b_0\right)_{-2}$.

Теоретическая справка

То, каким образом мы выполняем арифметические действия, тесно связано с тем, каким образом мы представляем числа, с которыми работаем; поэтому наше изучение арифметики естественно начать с обсуждения принципиальных способов представления чисел.

Позиционное представление с основанием (или по основанию) ***b*** определяется правилом

$$\left(\ldots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \ldots\right)_b = \ldots + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \ldots$$

например, $(520.3)_6 = 5 \cdot 6^2 + 2 \cdot 6^1 + 0 + 3 \cdot 6^{-1} = 192\frac{1}{2}$. Наша традиционная десятичная система счисления — это, разумеется, тот частный случай, когда ***b*** равно десяти и когда значения ***a*** выбираются из “десятичных цифр”

0, 1, 2, 3, 4, 5, 6, 7, 8, 9;

в этом случае индекс ***b*** можно опускать.

Простейшее обобщение десятичной системы счисления получается, когда в качестве ***b*** берут любое целое число, большее единицы, и числа ***a*** — это целые числа из интервала $0 \le a_k < b$.

Так получаются стандартные двоичная (***b***=2), троичная (***b***=3), четверичная (***b***=4), пятеричная (***b***=5) системы счисления. Вообще, в качестве ***b*** можно было бы взять произвольное число, а числа ***a*** выбирать из произвольного заранее заданного множества чисел; как мы увидим, это приводит к некоторым интересным ситуациям.

Точка, стоящая между $\boldsymbol{a_0}$ и $\boldsymbol{a_{-1}}$, называется позиционной (или разделительной) точкой. (В случае $\boldsymbol{b}$=10 ее называют также десятичной точкой, в случае $\boldsymbol{b}$=2 — двоичной точкой и т. д.) В европейских странах вместо позиционной точки часто используют запятую (в т.ч., в России; точка используется в США).

Числа $\boldsymbol{a}$ называют цифрами представления. Цифру $\boldsymbol{a_k}$ с большим $\boldsymbol{k}$ называют "более значимой", чем цифру $\boldsymbol{a_k}$, с меньшим $\boldsymbol{k}$; соответственно самую левую (ведущую, или головную) цифру называют наиболее значимой цифрой, а самую правую (хвостовую) — наименее значимой. В стандартной двоичной системе двоичные цифры часто называют битами. В стандартной шестнадцатеричной системе (с основанием шестнадцать) шестнадцатеричные цифры от нуля до пятнадцати обозначают обычно так:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E. F.

Имеется много интересных вариантов позиционных систем счисления, помимо стандартных n-арных систем, обсуждавшихся до сих пор. Например, мы могли бы рассматривать числа по основанию (—10), так что

$$(\ldots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} \ldots)_{-10} =$$
$$= \ldots + a_{3.}(-10)^3 + a_2(-10)^2 + a_1(-10)^{-1} + a_0 + a_{-1}(-10)_{-1} + a_{-2}(-10)^{-2} + \ldots =$$
$$= \ldots - 1000a_3 + 100a_2 - 10a_1 + a_0 - \frac{1}{10}a_{-1} + \frac{1}{100}a_{-2} - \ldots$$

Здесь, как и в обычной десятичной системе, цифры a_k удовлетворяют неравенствам $0 \le a_k \le 9$. Число 1234567890 запишется в такой «нега-десятичной» системе в виде

$$\mathbf{(1\ 93755\ 73910)_{-10}}$$

так как оно равно как раз 10305070900—9070503010. Интересно отметить, что его знаковое обращение, отрицательное число —1234567890, записывается в виде

$$\mathbf{(28466\ 48290)_{-10}}$$

и в действительности любое вещественное число, положительное или отрицательное, может быть представлено в системе по основанию —10 без знака.

Преобразование из двоичной системы в десятичную — одна из наиболее машинно-зависимых операций, поскольку инженеры постоянно изобретают различные способы реализации этой операции в аппаратуре компьютера.

Будем считать, что преобразованию подлежат только неотрицательные числа, так как манипуляцию со знаками учесть легко.

Предположим, что выполняется преобразование из основания ***b*** в основание ***B***. В основе большинства программ преобразования из одного основания в другое лежат операции умножения и деления, которые выполняются по одной из следующих четырех схем.

1. Деление на ***B*** при помощи арифметических действий над величинами с позиционным представлением по основанию ***b***. Дано целое число ***u***. Его представление $(U_M \ldots U_1 U_0)_B$ по основанию ***B*** получаем следующим образом:

$$
\begin{aligned}
&U_0 = u \bmod B,\\
&U_1 = \lfloor u/B \rfloor \bmod B,\\
&U_2 = \lfloor \lfloor u/B \rfloor / B \rfloor \bmod B,\\
&\ldots
\end{aligned}
$$

и т. д., пока не получим $\lfloor \ldots \lfloor \lfloor u/B \rfloor / B \rfloor \ldots / B \rfloor = 0$.

Например $10_{10} = 1010_2$:

$$
\begin{aligned}
&\lfloor \lfloor \lfloor 10/2 \rfloor /2 \rfloor /2 \rfloor \bmod 2 = 1;\\
&\lfloor \lfloor 10/2 \rfloor /2 \rfloor \bmod 2 = 0;\\
&\lfloor 10/2 \rfloor \bmod 2 = 1;\\
&10 \bmod 2 = 0.
\end{aligned}
$$

2. Умножение на ***b*** при помощи арифметики основания ***B***. Если представление числа ***u*** по основанию ***b*** имеет вид $(u_m \ldots u_1 u_0)_b$, то мы можем, воспользовавшись арифметикой основания ***B***, вычислить многочлен $u_m b^m + \ldots + u_1 b + u_0 = u$ в виде

$$((\ldots(u_m b + u_{m-1})b + \ldots)b + u_1)b + u_0 .$$

Например $1010_2 = 10_{10}$:

$((1 \cdot 2 + 0) \cdot 2 + 1) \cdot 2 + 0 = 10$.

Часто бывает невозможно точно выразить конечную дробь с основанием ***b*** как конечную дробь с основанием ***B***. Например, дробь $1/10$ имеет бесконечное двоичное представление $(0.0001100110011\ldots)_2$. Поэтому применяются различные методы округления результата до *M* знаков.

3. Умножение на $\boldsymbol{B}$ при помощи арифметики основания $\boldsymbol{b}$. Дано дробное число $\boldsymbol{u}$; мы получаем последовательные цифры его представления $(.U_{-1}U_{-2}...U_{-M})_B$ по основанию $\boldsymbol{B}$ следующим образом:

$$\begin{aligned}
&U_{-1} = \lfloor uB \rfloor,\\
&U_{-2} = \lfloor \{uB\}B \rfloor,\\
&U_{-3} = \lfloor \{\{uB\}B\}B \rfloor,\\
&\ldots
\end{aligned}$$

где $\{x\}$ обозначает $x \bmod 1 = x - \lfloor x \rfloor$. В случае, когда нужно округлить результат до M знаков, можно прекратить вычисление, как только найдено U_{-M}.

Например $(0.7)_{10} = (0.1011...)_2$:

$$\begin{aligned}
&\lfloor 0.7 \cdot 2 \rfloor = 1;\\
&\lfloor \{0.7 \cdot 2\} \cdot 2 \rfloor = 0;\\
&\lfloor \{\{0.7 \cdot 2\} \cdot 2\} \cdot 2 \rfloor = 1;\\
&\lfloor \{\{\{0.7 \cdot 2\} \cdot 2\} \cdot 2\} \cdot 2 \rfloor = 1;\\
&\ldots
\end{aligned}$$

4. Деление на $\boldsymbol{b}$ при помощи арифметики основания $\boldsymbol{B}$. Если представление числа $\boldsymbol{u}$ по основанию $\boldsymbol{b}$ имеет вид $(0.u_{-1}u_{-2}...u_{-m})_b$, то можно, используя арифметику основания $\boldsymbol{B}$, вычислить $u_{-1}b^{-1} + u_{-2}b^{-2} + ... + u_{-m}b^{-m}$ в виде

$$\left(\left(...\left(u_{-m}/b + u_{1-m}\right)/b + ... + u_{-2}\right)/b + u_{-1}\right)/b. \quad (1.6)$$

Например $(0.1011)_2 = (0.6875)_{10} \approx (0.7)_{10}$:

$$\left(\left(\left(1/2 + 1\right)/2 + 0\right)/2 + 1\right)/2.$$

В результате усечения или округления при делении на $\boldsymbol{b}$ могут появиться ошибки. Они обычно малы, но не всегда.

В ЭВМ используется двоичная система счисления. Числа в ней представляются очень большим количеством разрядов. В восьмеричной же и шестнадцатеричной системах числа читаются почти так же легко, как десятичные, требуют соответственно в три (восьмеричная) и в четыре (шестнадцатеричная) раза меньше разрядов, чем в двоичной системе (ведь числа 8 и 16 — соответственно, третья и четвертая степени числа 2). Это можно обобщить на произвольную степень m некоторого основания системы n.

В таких случаях пользуются простым алгоритмом. Рассмотрим его на примере систем по основанию 2, 8, 16:

1. Перевод восьмеричных и шестнадцатеричных чисел в двоичную систему очень прост: достаточно каждую цифру заменить эквивалентной ей двоичной триадой (тройкой цифр) или тетрадой (четверкой цифр).

2. Чтобы перевести число из двоичной системы в восьмеричную или шестнадцатеричную, его нужно разбить влево и вправо от запятой на триады (для восьмеричной) или тетрады (для шестнадцатеричной) и каждую такую группу заменить соответствующей восьмеричной (шестнадцатеричной) цифрой.

Текст программы

Далее представлен текст программы на языке C++, реализующей преобразование числа из обратного двоичного кода в систему счисления с основанием -2.

```
#include <iostream>
#include <string>
using namespace std;
struct neg_bin{
    bool carry;
    int num;
};

bool sign_bit(int n){
    return n>>31;
};
void print_bin(int n){
    for(int i=0;i<32;i++)
    {
        cout<<(sign_bit(n)?'1':'0');
        n<<=1;
    };
    cout<<endl;
}
neg_bin bin2neg_bin(int n){
    bool cnt=false,carry=false;
    if(sign_bit(n)){
        cnt=true;
        n=~n;
    }
    int tmp=0,i=0;
    bool bit;
    while(n){
        cnt=!cnt;
        bit=n&1;
        if(carry){
            if(cnt&&!bit)
                carry=false;
            bit=!bit;
        } else {
            carry=bit&&!cnt;
```

```
        }
        tmp+=(bit?1<<i:0);
        n>>=1;
        i++;
    }
    neg_bin res;
    res.num=tmp;
    bool carry0=carry;
    bool carry1=carry&&cnt;
    res.num+=(carry0?1<<i:0);
    if(i>=31)
        res.carry=true;
    else{
        res.carry=false;
        res.num+=carry1?1<<(i+1):0;
    }
    return res;
}
int neg_bin2dec(neg_bin b){
    int i=0;
    int t=0,a=0;
    while(b.num){
        a=((b.num&1)*(1<<i));
        if(i%2)
            a=-a;
        t+=a;
        b.num>>=1;
        i++;
    }
    return t;
}
int main(){
    int n;
    cout<<"Введите целое число: ";
    cin>>n;
    if(n<0)
        n--;
    cout<<"Обратный код числа: ";
    print_bin(n);
    neg_bin b=bin2neg_bin(n);
    cout<<"Представление числа в системе счисления с основанием -2: "<<endl;
    if(b.carry)
        cout<<'1';
    print_bin(b.num);
    return 0;
}
```

Тестовый пример

На рисунке 1 представлен пример работы программы, реализующей преобразование числа из обратного двоичного кода в систему счисления с основанием -2.

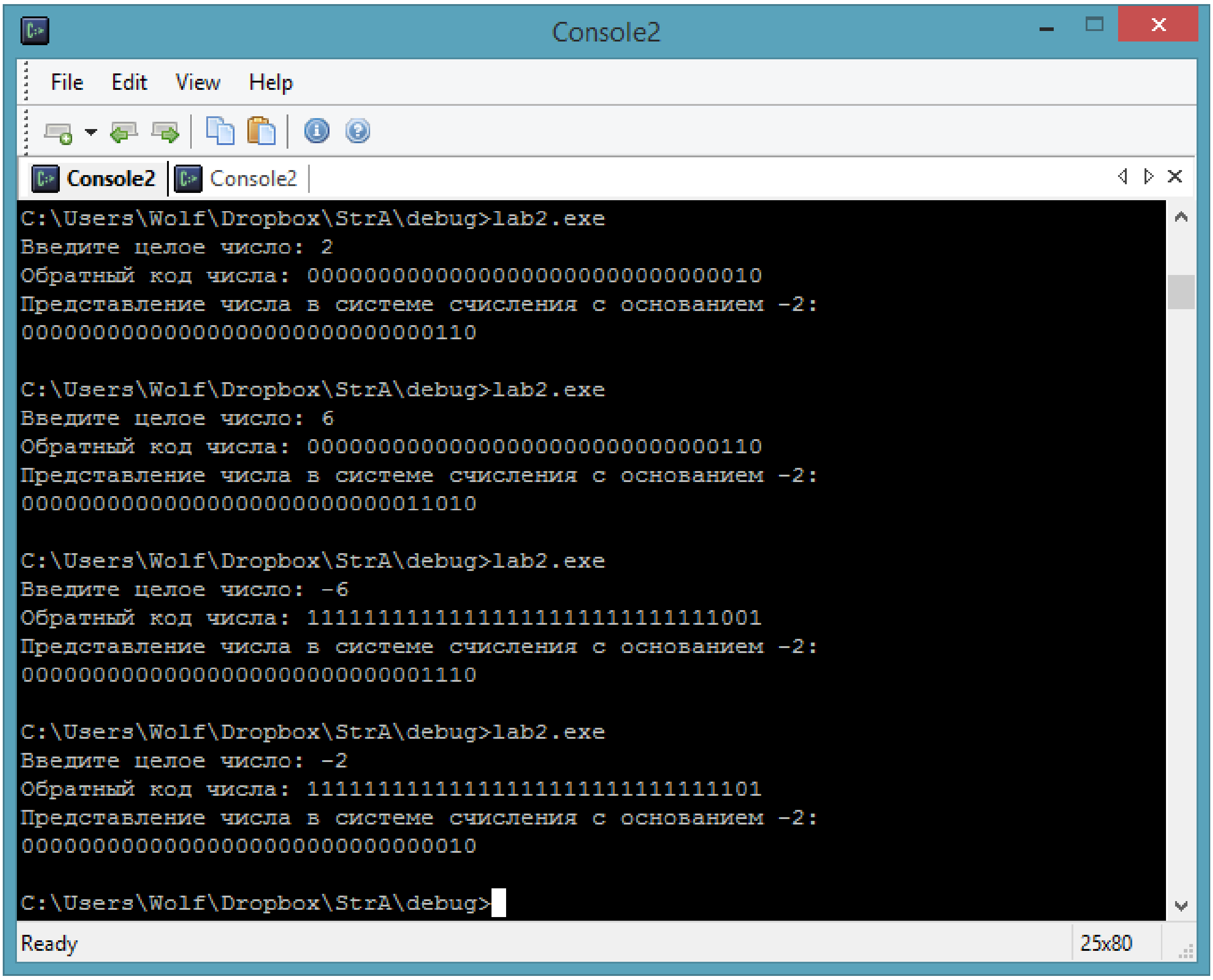

Рисунок 1— Пример работы программы перевода чисел в систему счисления с основанием -2

Вывод

В данной работе я познакомился с математическими основами позиционных систем счисления и методами перевода чисел между различными системами счисления. Была написана программа преобразования чисел в систему счисления с основанием -2. Преобразование чисел между различными системами счисления очень частая операция, поскольку люди оперируют десятичной, а вычислительные машины — двоичной системой счисления.