**Лабораторна робота №1**

**Базові системи числення**

**Мета роботи**: закріплення знань про системи числення та перетворення між ними; оволодіння навичками складання програм з перетворення чисел в різні позиційні системи числення.

*Стислі теоретичні відомості*

**Система числення** – сукупність правил запису чисел цифровими знаками.

**Алфавіт системи числення** – сукупність символів, що використовуються для запису числа в даній системі числення.

Системи числення поділяються на 3 групи:

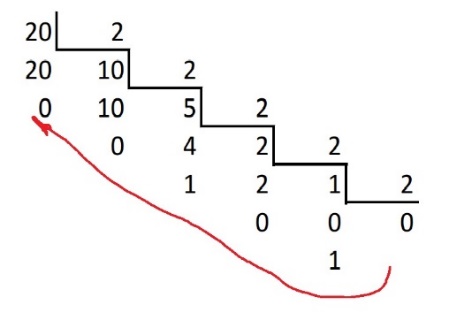
**Непозиційні** – системи числення у яких значення числа визначається сумою цифр у ньому. Найрозповсюдженим прикладом такої системи числення є римська.

**Позиційні** – системи числення у яких значення цифри залежить від її місця. Наприклад, звичайна десяткова система числення, двійкова, вісімкова, шістнадцяткова…

*Переведення у десяткову систему числення:*

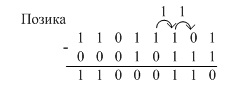
Ap = anpn + an-1pn-1 +...+ a2p2 + a1p1 + a0p0 + a-1p-1 + a-2p-2 + ... + a-kp-k

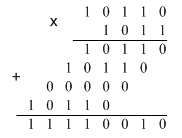
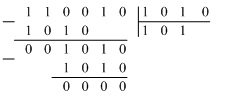
*Переведення з десяткової системи числення у потрібн*у (переведення цілої і дробної частини):

Де 2 – потрібна система числення.

*Додавання, віднімання, множення та ділення чисел у різних системах числення* виконуються так само як і в десятковій системі числення:

**Завдання**

1. Розробіть мовою програмування C/С++ функції перетворення числа з однієї системи числення в іншу відповідно до варіанту (табл. 1.1). Числа задавати та отримувати у вигляді рядка.

2. Розробіть функції для додавання та віднімання чисел в даних системах числення.

|  |  |
| --- | --- |
| 0101 | 2 ⇒ 16 |

*Розв’язання*

1. **Текст програми**

//- - - main.cpp - - -

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

#include "Num.h"

using namespace std;

typedef unsigned short int usi;

template <typename T>

T prompt(const string label = "Введите значение: ");

usi askRadix(const string label = "Введите основание СЧ: ");

void assert(string val, string rightVal, string description);

void help(), cls(), add(), sub(), mul(), div();

void leftShift(), compare(), convert(), runTests();

int main() {

   setlocale(LC\_ALL, "Russian");

   system("color F0");

   string prevCommand = "help";

   help();

   while (true) {

      string command = prompt<string>("> ");

      if (command == "0") return 0;

      if (command == "1") {

         cout << "> " << prevCommand << endl;

         runCommad(prevCommand);

      }

      else {

         prevCommand = command;

         runCommad(command);

      }

   }

   return 0;

}

void runCommad(const string command) {

   if (command == "add") add();

   else if (command == "sub") sub();

   else if (command == "mul") mul();

   else if (command == "div") div();

   else if (command == "leftShift") leftShift();

   else if (command == "compare") compare();

   else if (command == "convert") convert();

   else if (command == "runTests") runTests();

   else if (command == "help") help();

   else if (command == "cls") cls();

   else cout << "Команда не найдена" << endl;

}

void help() {

   cout << "- - - Number System 0.0.1 - - -" << endl << endl

      << "> add: Cуммирование (+)" << endl

      << "> sub: Вичитание (-)" << endl

      << "> mul: Умножение (\*)" << endl

      << "> div: Деление (/)" << endl

      << "> leftShift: Сдвиг влево (<<)" << endl

      << "> compare: Сравнить числа" << endl

      << "> convert: Изменить систему счисления" << endl

      << "> runTests: Запустить тесты" << endl << endl

      << "> help: помощь" << endl

      << "> cls: очистить консоль" << endl

      << "> 1: повторить последнюю команду" << endl

      << "> 0: Выход" << endl << endl;

}

void add() {

   usi radix = askRadix();

   Num num1(prompt<string>("Введите первое число: "), radix);

   Num num2(prompt<string>("Введите второе число: "), radix);

   cout << endl << num1 << " + " << num2 << ": " << endl;

   printInColor(num1 + num2, Green); cout << endl << endl;

}

void sub() {

   usi radix = askRadix();

   Num num1(prompt<string>("Из числа: "), radix);

   Num num2(prompt<string>("Вычесть: "), radix);

   cout << endl << num1 << " - " << num2 << ": " << endl;

   printInColor(num1 - num2, Green); cout << endl << endl;

}

void mul() {

   usi radix = askRadix();

   Num num1(prompt<string>("Первое число: "), radix);

   Num num2(prompt<string>("Второе число: "), radix);

   cout << endl << num1 << " \* " << num2 << ": " << endl;

   printInColor(num1 \* num2, Green); cout << endl << endl;

}

void div() {

   usi radix = askRadix();

   Num num1(prompt<string>("Число: "), radix);

   Num num2(prompt<string>("Разделить на: "), radix);

   cout << endl << num1 << " / " << num2 << ": " << endl;

   printInColor(num1 / num2, Green); cout << endl << endl;

}

void leftShift() {

   usi radix = askRadix();

   Num num(prompt<string>("Введите число: "), radix);

   unsigned shift = min(prompt<unsigned>("Сместить на: "), 1000);

   cout << endl << num << " << " << shift << ": " << endl;

   printInColor(num << shift, Green); cout << endl << endl;

}

void compare() {

   usi radix1 = askRadix("Введите основание СЧ первого числа: ");

   Num num1(prompt<string>("Первое число: "), radix1);

   usi radix2 = askRadix("Введите основание СЧ второго числа: ");

   Num num2(prompt<string>("Второе число: "), radix2);

   string res = "Такое же как число";

   if (num1.compare(num2) == 1) res = "Больше чем число";

   else if (num1.compare(num2) == -1) res = "Меньше чем число";

   cout << endl << "Число " << num1 << "(" << num1.getRadix() << ")" << endl;

   printInColor(res, Green);

   cout << ":" << endl << num2 << "(" << num2.getRadix() << ")" << endl << endl;

}

void convert() {

   usi radix1 = askRadix("Введите текущее основание СЧ: ");

   Num num(prompt<string>("Введите число: "), radix1);

   usi radix2 = askRadix("Введите новое основание СЧ: ");

   cout << endl << num << "(" << radix1 << ")" << " -> " << endl;

   num.setSystem(radix2);

   string res = num.toString() + "(" + to\_string(num.getRadix()) + ")";

   printInColor(res, Green); cout << endl << endl;

}

void cls() { system("cls"); }

usi askRadix(const string label) {

   usi radix = 0;

   while (radix < 2 || radix > 36) radix = prompt<usi>(label);

   return radix;

}

void runTests() {

   cout << "Add (+):" << endl;

   assert(Num("0", 10) + Num("0", 10), "0", "0 + 0 = 0 (10)");

   assert(Num("10", 10) + Num("20", 10), "30", "10 + 20 = 30 (10)");

   //...

}

// - - - Num.h - - -

#pragma once

class Num {

private:

   usi radix = 2; //Основание системы счисления

   vector<usi> bits; //Биты числа (bits[0] - младший бит)

   void setRadix(usi radix);

   usi getCorrectReadix(usi radix);

   usi charToDigit(char c);

   char digitToChar(usi d);

   Num mulOnDigit(usi digit);

public:

   Num(string num, usi radix = 10);

   Num(unsigned long long num, usi radix = 10);

   Num(Num \*num);

   void parse(string num, usi radix = 10);

   void parse(Num &num);

   Num plus(Num &\_num);

   Num minus(Num &\_num);

   Num mul(Num &\_num);

   Num div(Num &\_num);

   Num leftShift(unsigned int n);

   short int compare(Num &\_num);

   void setSystem(usi \_radix);

   usi getBit(usi pos);

   void setBit(usi pos, usi val);

   void addToBit(usi pos, usi val);

   usi getRadix();

   unsigned int size();

   Num getCorrectNum(Num &num);

   Num copy();

   void clean();

   string toString();

   unsigned long long toReal();

   friend Num operator + (Num num1, Num num2) { return num1.plus(num2); }

   friend Num operator - (Num num1, Num num2) { return num1.minus(num2); }

   friend Num operator \* (Num num1, Num num2) { return num1.mul(num2); }

   friend Num operator / (Num num1, Num num2) { return num1.div(num2); }

   friend Num operator << (Num num, int shift) { return num.leftShift(shift); }

   friend ostream& operator << (ostream &out, Num &num) { return out << num.toString(); }

   operator string() { return toString(); }

};

// - - - Num.cpp - - -

#include "Num.h"

void Num::setRadix(usi radix) {

   this->radix = getCorrectReadix(radix);

}

usi Num::getCorrectReadix(usi radix) {

   if (radix < 2) radix = 2;

   else if (radix > 36) radix = 36;

   return radix;

}

usi Num::charToDigit(char c) {

   c = toupper(c);

   if (c >= '0' && c <= '9') return c - '0';

   if (c >= 'A' && c <= 'Z') return 10 + (c - 'A');

   return 0;

}

char Num::digitToChar(usi d) {

   if (d >= 0 && d <= 9) return d + '0';

   if (d >= 10 && d <= 35) return (d + 'A') - 10;

   return '\*';

}

//Умножение на одну цифру

Num Num::mulOnDigit(usi digit) {

   Num num = copy();

   usi radix = num.getRadix();

   if (digit >= radix) digit = radix - 1;

   int len = num.size();

   int rest = 0;

   for (int i = 0; i < len; i++) {

      int mul = num.getBit(i) \* digit + rest;

      num.setBit(i, mul % radix);

      rest = mul / radix;

   }

   if (rest) num.setBit(len, rest);

   return num;

}

Num::Num(string num, usi radix) {

   parse(num, radix);

}

Num::Num(unsigned long long num, usi radix) {

   parse(to\_string(num), 10);

   setSystem(radix);

}

Num::Num(Num \*num) { parse(\*num); }

void Num::parse(string num, usi radix) {

   if (num == "") num = "0";

   bits.clear();

   setRadix(radix);

   for (int i = num.length() - 1; i >= 0; i--) {

      usi bit = min(charToDigit(num[i]), getRadix() - 1);

      bits.push\_back(bit);

   }

   clean();

}

Num Num::plus(Num &\_num) {

   Num num1 = copy(), num2 = getCorrectNum(\_num);

   int len1 = num1.size(), len2 = num2.size();

   usi radix = num1.getRadix();

   for (int i = 0; i < len1 || i < len2; i++) {

      usi sum = num1.getBit(i) + num2.getBit(i);

      num1.setBit(i, sum % radix);

      num1.addToBit(i + 1, sum / radix);

   }

   return num1;

}

Num Num::minus(Num &\_num) {

   Num num1 = copy(), num2 = getCorrectNum(\_num);

   int len1 = num1.size(), len2 = num2.size();

   usi radix = num1.getRadix();

   if (len1 < len2) return Num(0, radix);

   for (int i = 0; i < len1; i++) {

      short sub = num1.getBit(i) - num2.getBit(i); //sub может быть отрицательным

      if (sub < 0) {

         sub += radix; //Окончательное значение текущего бита

         int notZiro = -1;

         //Ищем бит, у которого можно взять 1

         for (int j = i + 1; j < len1; j++) {

            if (num1.getBit(j) != 0) {

               num1.addToBit(j, -1);

               notZiro = j;

               break;

            }

         }

         if (notZiro == -1) return Num(0, radix);

         //Добавляем по 1 всем промежуточным битам

         for (int j = notZiro - 1; j > i; j--) {

            num1.setBit(j, radix - 1);

         }

      }

      num1.setBit(i, sub);

   }

   return num1;

}

Num Num::mul(Num &\_num) {

   Num num1 = copy(), num2 = getCorrectNum(\_num);

   int len1 = num1.bits.size(), len2 = num2.bits.size();

   usi radix = num1.getRadix();

   Num res(0, radix);

   for (int i = 0; i < len2; i++) {

      res = res + (num1.mulOnDigit(num2.getBit(i)) << i);

   }

   return res;

}

Num Num::div(Num &\_num) {

   Num num1 = copy(), num2 = getCorrectNum(\_num);

   usi radix = num1.getRadix();

   if (Num(0, radix).compare(\_num) == 0) {

      return Num(0, radix);

   }

   Num res("0", radix), tmp("0", radix);

   int n = 1;

   while (num1.compare(num2) >= 0) {

      auto b = num1.bits.begin() + num1.size();

      tmp.bits = vector<usi>(b - n, b);

      if (tmp.compare(num2) == -1) {

         n++;

         continue;

      }

      Num next(0, radix), k(0, radix);

      while (true) {

         next = next + num2;

         if (next.compare(tmp) > 0) {

            next = next - num2;

            break;

         };

         k = k + Num(1);

      }

      int shift = num1.size() - tmp.size();

      num1 = num1 - (next << shift);

      res = res + (k << shift);

      n = 1;

   }

   return res;

}

Num Num::leftShift(unsigned int n) {

   Num num = copy();

   vector<usi> shift(n, 0);

   num.bits.insert(num.bits.begin(), shift.begin(), shift.end());

   return num;

}

//Сравнивает данное число c переданным

//Если оно больше то возв.: 1, меньше: -1, одинаковые: 0

short int Num::compare(Num &\_num) {

   Num num = getCorrectNum(\_num);

   int len1 = size(), len2 = num.size();

   if (len1 > len2) return 1;

   else if (len1 < len2) return -1;

   for (int i = len1 - 1; i >= 0; i--) {

      if (getBit(i) > num.getBit(i)) return 1;

      else if (getBit(i) < num.getBit(i)) return -1;

   }

   return 0;

}

void Num::setSystem(usi \_radix) {

   if (this->radix == \_radix) return;

   \_radix = getCorrectReadix(\_radix);

   Num radix(\_radix), curRadix(getRadix()), all("0", 10);

   //Calc RealNum

   if (getRadix() == 10) {

      all.parse(\*this);

   }

   else {

      for (int i = size() - 1; i >= 0; i--) {

         all = (all \* curRadix) + Num(getBit(i));

      }

   }

   //To result

   string res = "";

   while (all.compare(radix) >= 0) {

      Num div = all / radix;

      usi rem = usi((all - (div \* radix)).toReal());

      res = digitToChar(rem) + res;

      all = div;

   }

   res = digitToChar(usi(all.toReal())) + res;

   parse(\*new Num(res, \_radix));

}

string Num::toString() {

   string strNum = "";

   for (int i = size() - 1; i >= 0; i--) {

      strNum += digitToChar(bits[i]);

   }

   return strNum;

}

//Возвращает реальное число

//Число не должно быть слишком большим

unsigned long long Num::toReal() {

   unsigned long long all = 0;

   for (int i = size() - 1; i >= 0; i--) {

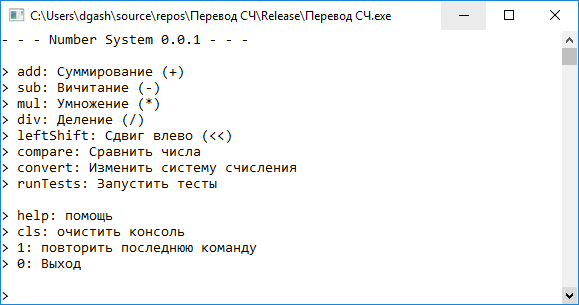
      all = all \* getRadix() + getBit(i);

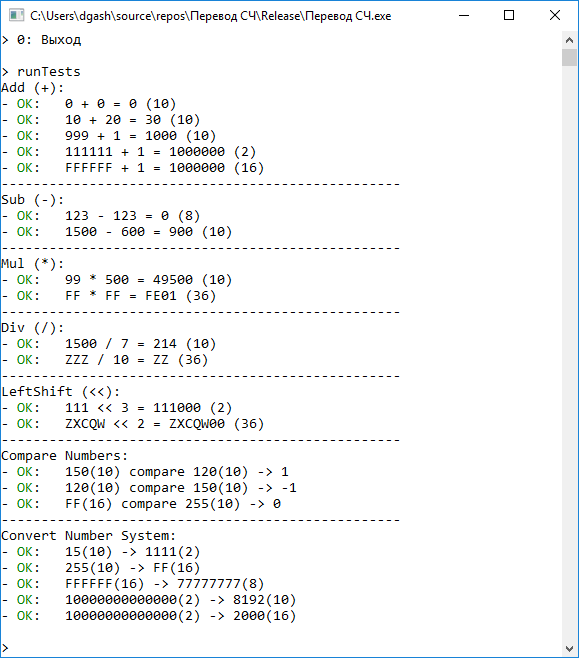
   }

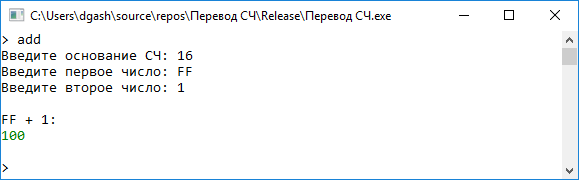
   return all;

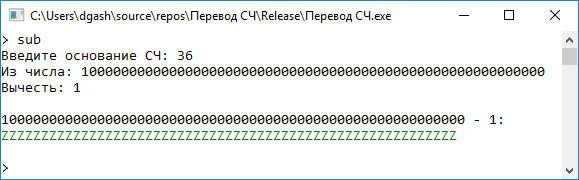
}

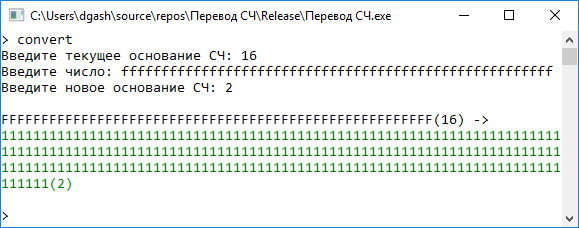
1. Результат виконання програми:

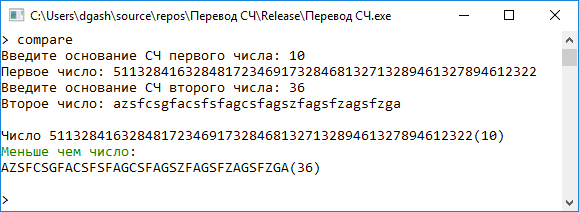












**Висновок:** на цій лабораторній роботі я закріпив знання про системи числення та перетворення між ними; оволодів навичками складання програм з перетворення чисел в різні позиційні системи числення.