**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

Катедра: **Компютърни системи и технологии**

КУРСОВА РАБОТА  
ПО СИНТЕЗ И АНАЛИЗ НА АЛГОРИТМИ

Студент:

Факултетен номер:

Група:

Специалност:

Дата: Изготвил:  
 Проверил:

**1. Задание.**

**15.** Дадени са две големи цели числа N1 и N2 (до 70 разряда).Да се напишат програмни фрагменти за:

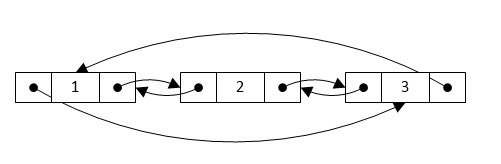
1. представяне на числата чрез списъци;
2. сравнение на двете числа (чрез използване на списъците) и отпечатване на резултата;
3. умножение на голямо цяло число с едноцифрено число;
4. сумиране на две големи цели числа;
5. отпечатване стойността на израза 2\*N1+3\*N2, като се използват създадените в подточки c) и d) подпрограми.

**2. Описание.**

*(Среда за програмиране – Microsoft Visual Studio Professional 2013)*

За представяне на големите цели числа се използва двусвързан цикличен списък. Това дава възможност за обхождане на списъка в две посоки, което е необходимо при извършване на операциите умножение (от последния до първия), сумиране (от последния до първия), сравняване (от първия до последния) и извеждане на списъка на конзолата (от първия до последния).

За създаване на списъка се използва функцията CreateBigInteger(). Тъй като числата са много големи (до 70 разряда) се налага въвеждането на числата в масив от символи. Този масив от символи се подава към функцията CreateBigInteger(), заедно с „главата“ на списъка (като В/И променлива). На фиг. 1 е представено едно цяло число чрез двусвързан списък.



Фигура 1 Представяне на числото 123 в двусвързан цикличен списък

За извеждане на списъка на конзолата се използва функцията PrintNumber(). Извеждането става чрез един цикъл do-while, с помощта, на който списъкът се обхожда от първия до последния елемент, използвайки указателя next.

Сравняването на две големи числа става чрез функцията Compare(). Тя връща 0 ако двете числа са равни, -1 – ако първото числа е по-малко от второто, 1 – ако първото число е по-голямо от второто и 2 – ако единият или и двата списъка имат стойност NULL, т.е е празен(са празни).

Умножението на голямо цяло число с едноцифрено числo става чрез функцията Multiplication(). Първоначално резултатът се съхранява в стринг (*т.к не е уточнено по какъв начин да стане това, избрах да използвам стринг, а и имаме вече функция за прехвърляне на число от низ в списък. Но по-добър вариант може би е да се използва стек или направо структурата за двусвързания списък*), който се подава към функцията за създаване на двусвързан списък, заедно с „главата“ на списъка, в който ще бъде прехвърлено числото. „Главата“ на този списък се връща като резултат от умножението.

Сумирането на две големи числа става чрез функцията Sum(). И тук резултатът първоначално се съхранява в стринг, а след това се прехвърля в списък. Този списък се връща като резултат от сумирането.

В функциите Sum() и Multiplication() се използва вградената функция \_itoa\_s(), която се използва за конвертиране на цяло число в масив от символи.

**3. Описание на използваните функции**

1. Вид на функцията: **int main();**

Действие: **главна функция;**  
 Параметри: **няма;**  
 Използвани глобални променливи: **BigInteger firstNumber, secondNumber;**  
 Извиквани функции: **CreateBigInteger(), PrintNumber(), Compare(), Multiplication(), Sum().**

1. Вид на функцията: **void CreateBigInteger(BigInteger &numberList, const char \*strNumber);**

Действие: **Подпрограма за създаване на дълго цяло положително число;**  
 Параметри: **BigInteger &numberList – списък (указател към главата на списъка);**

**const char \*strNumber – числото за прехвърляне в списъка;**  
 Използвани глобални променливи: **няма;**  
 Извиквани функции: **няма;**

1. Вид на функцията: **void PrintNumber(BigInteger number);**

Действие: **Подпрограма за извеждане на списъка на конзолата;**  
 Параметри: **BigInteger number – списък (указател към главата на списъка);**  
 Използвани глобални променливи: **няма;**  
 Извиквани функции: **няма;**

1. Вид на функцията: **int Compare(BigInteger firstNum, BigInteger secondNum);**

Действие: **Подпрограма за сравняване на две големи цели числа;**  
 Параметри: **BigInteger firstNum - списък (указател към главата на първия списък); BigInteger secondNum - списък (указател към главата на втория списък);**  
 Използвани глобални променливи: **няма;**  
 Извиквани функции: **няма;**

1. Вид на функцията: **BigInteger Multiplication(unsigned short intNumber, BigInteger bigNumber);**

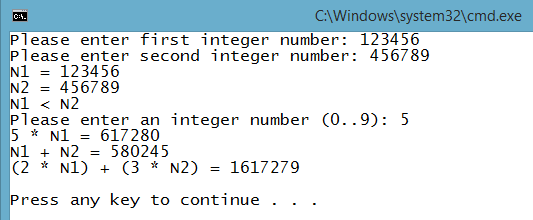
Действие: **Подпрограма за умножение на голямо цяло число с едноцифрено цяло число;**  
 Параметри: **unsigned short intNumber – едноцифреното цяло число;**

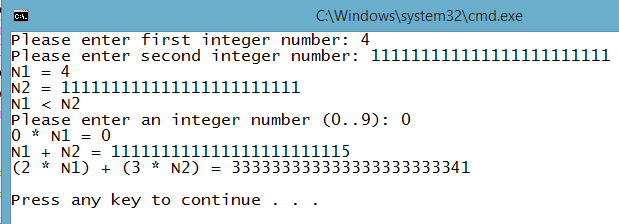
**BigInteger bigNumber - списък (указател към главата на списъка);**  
 Използвани глобални променливи: **няма;**  
 Извиквани функции: **CreateBigInteger();**

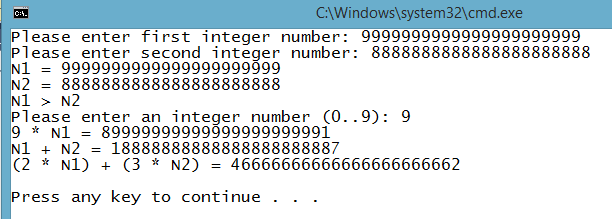
1. Вид на функцията: **BigInteger Sum(BigInteger firstNum, BigInteger secondNum);**

Действие: **Подпрограма за събиране на две големи цели числа;**  
 Параметри: **BigInteger firstNum - списък (указател към главата на първия списък); BigInteger secondNum - списък (указател към главата на втория списък);**  
 Използвани глобални променливи: **няма;**  
 Извиквани функции: **CreateBigInteger();**

**4. Тестови примери**

****

****

****

**5. Код на програмата**

#include <iostream>

using namespace std;

typedef struct Number

{

unsigned short digit;

Number \*prev, \*next;

} \*BigInteger;

BigInteger firstNumber, secondNumber;

void CreateBigInteger(BigInteger &numberList, const char \*strNumber)

{

BigInteger p, last;

numberList = new Number;

numberList->digit = (unsigned short)strNumber[0] - 48;

numberList->next = numberList;

numberList->prev = numberList;

p = new Number;

for (int i = 1; i < strlen(strNumber); i++, p = p->next)

{

if (i == 1)

{

numberList->next = p;

numberList->prev = p;

p->prev = numberList;

p->next = NULL;

}

last = new Number;

p->digit = (unsigned short)strNumber[i] - 48;

p->next = last;

last->prev = p;

if (i == (strlen(strNumber) - 1))

{

p->next = numberList;

numberList->prev = p;

}

}

}

void PrintNumber(BigInteger number)

{

if (number)

{

BigInteger q = number;

do

{

cout << q->digit;

q = q->next;

}

while (q != number);

}

cout << endl;

}

int Compare(BigInteger firstNum, BigInteger secondNum)

{

BigInteger f = firstNum;

BigInteger s = secondNum;

if (f && s)

{

int compareDigit = 0;

do

{

if (compareDigit == 0)

{

if (f->digit > s->digit)

{

compareDigit = 1;

}

else if (f->digit < s->digit)

{

compareDigit = -1;

}

else

{

compareDigit = 0;

}

}

f = f->next;

s = s->next;

}

while (f != firstNum && s != secondNum);

if (f == firstNum && s == secondNum)

{

return compareDigit;

}

else if (f == firstNum && s != secondNum)

{

return -1;

}

else

{

return 1;

}

}

return 2;

}

BigInteger Multiplication(unsigned short intNumber, BigInteger bigNumber)

{

if (bigNumber)

{

BigInteger result = NULL, q = bigNumber;

string strResult = "";

char charDigit[2];

int buffer, carry = 0;

if (intNumber != 0)

{

do

{

q = q->prev;

buffer = q->digit \* intNumber + carry;

if (buffer > 9)

{

\_itoa\_s(buffer % 10, charDigit, 10);

carry = buffer / 10;

}

else

{

\_itoa\_s(buffer, charDigit, 10);

carry = 0;

}

strResult = charDigit + strResult;

}

while (q != bigNumber);

if (carry != 0)

{

\_itoa\_s(carry, charDigit, 10);

strResult = charDigit + strResult;

}

}

else

{

strResult = "0";

}

CreateBigInteger(result, strResult.c\_str());

return result;

}

return NULL;

}

BigInteger Sum(BigInteger firstNum, BigInteger secondNum)

{

if (firstNum && secondNum)

{

int carry = 0;

int buffer;

char charDigit[2];

string strResult = "";

BigInteger result = NULL;

BigInteger f = firstNum;

BigInteger s = secondNum;

f = f->prev;

s = s->prev;

buffer = f->digit + s->digit;

carry = (buffer > 9) ? 1 : 0;

\_itoa\_s(buffer % 10, charDigit, 10);

strResult = charDigit;

while (f != firstNum || s != secondNum)

{

if (f != firstNum && s != secondNum)

{

f = f->prev;

s = s->prev;

buffer = f->digit + s->digit + carry;

}

else if (f == firstNum && s != secondNum)

{

s = s->prev;

buffer = s->digit + carry;

}

else if (f != firstNum && s == secondNum)

{

f = f->prev;

buffer = f->digit + carry;

}

carry = (buffer > 9) ? 1 : 0;

\_itoa\_s(buffer % 10, charDigit, 10);

strResult = charDigit + strResult;

}

if (carry == 1)

{

\_itoa\_s(carry, charDigit, 10);

strResult = charDigit + strResult;

}

CreateBigInteger(result, strResult.c\_str());

return result;

}

return NULL;

}

int main()

{

firstNumber = NULL;

secondNumber = NULL;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* READ INPUT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char strFirtsNumber[71];

char strSecondNumber[71];

cout << "Please enter first integer number: ";

cin >> strFirtsNumber;

// проверка дали не е въведен празен символ, защото потребителят има възможност

// да въвежда стойност null (\0) чрез клавишната комбинация Ctrl + Z.

if (strlen(strFirtsNumber) > 0)

{

CreateBigInteger(firstNumber, strFirtsNumber);

}

else

{

cout << "Invalid input!";

return 0;

}

cout << "Please enter second integer number: ";

cin >> strSecondNumber;

if (strlen(strSecondNumber) > 0)

{

CreateBigInteger(secondNumber, strSecondNumber);

}

else

{

cout << "Invalid input!";

return 0;

}

cout << "N1 = ";

PrintNumber(firstNumber);

cout << "N2 = ";

PrintNumber(secondNumber);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* COMPARE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int compareResult = Compare(firstNumber, secondNumber);

switch (compareResult)

{

case -1:

cout << "N1 < N2" << endl;

break;

case 0:

cout << "N1 = N2" << endl;

break;

case 1:

cout << "N1 > N2" << endl;

break;

default:

cout << "Some number not exist!" << endl;

break;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MULTIPLICATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned short intNum;

cout << "Please enter an integer number (0..9): ";

cin >> intNum;

cout << intNum << " \* N1 = ";

PrintNumber(Multiplication(intNum, firstNumber));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SUM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cout << "N1 + N2 = ";

PrintNumber(Sum(firstNumber, secondNumber));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* EXPRESSION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

cout << "(2 \* N1) + (3 \* N2) = ";

PrintNumber(Sum(Multiplication(2, firstNumber), Multiplication(3, secondNumber)));

cout << endl;

return 0;

}