Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»
Факультет информационных технологий и программирования

Программирование

Лабораторная работа №2

Выполнил студент: Шайдулин Михаил Андреевич
Группа: M3106

Санкт-Петербург
2021

1 uint1024_t.c

```c
#include <stdio.h>
#include <locale.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include "uint1024.h"
#include "uint1024_io.h"


int main() {
    uint1024_t x, y;

    printf("1: "); scanf_value(&x);

    printf("2: "); scanf_value(&y);

    printf("                :   "); printf_value(add_op(x, y));

    printf("\n");
    return 0;
}
```

2 uint1024.h

```
#include <stdio.h>
#include <locale.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>

typedef struct uint1024_t
{
    uint32_t numBlock[35];
} uint1024_t;

uint1024_t from_uint(unsigned int x)
{
    uint1024_t result;
    int filler = 0; // value to fill num blocks with
    memset(&result, filler, sizeof(uint32_t) * 35); // allocate memory for num
       blocks

    // fill num blocks with int32 values
    int i = 0;
    while ((x > 0) && (i < 35))
    {
        result.numBlock[i] = x % 1000000000;
        x = x / 1000000000; i++;

    }
    return result;
}

uint1024_t add_op (uint1024_t x, uint1024_t y)
{
    uint1024_t result; // variable to store result
    memset(&result, 0, sizeof(uint32_t) * 35); // set memory for num blocks

    for (int i = 0; i < 35; i++)
    {   // write to numBlock[i] value of sum
        result.numBlock[i] = (result.numBlock[i]) + (y.numBlock[i] + x.numBlock[
           i]) % 1000000000;
        if (i != 35 - 1) // remember the overflow value for next blocks
        { // only for blocks except last
            result.numBlock[1+i] = (y.numBlock[i] + x.numBlock[i]) / 1000000000
               ;
        }
    }
    return result;
}

uint1024_t subtr_op (uint1024_t x, uint1024_t y)
{
    uint1024_t result;
    uint64_t rest;

    int filler = 0; // filler value to fill blocks with
    memset(&result, filler, sizeof(uint32_t)*35); // allocate memory for num
       blocks

    int j;
    for (int i = 0; i < 35; i++)
    {
        if (x.numBlock[i] < y.numBlock[i])
        {
            rest = (1000000000 + x.numBlock[i] - y.numBlock[i]) % 1000000000;
            result.numBlock[i] = rest;
            if (i != 35-1)
```

```
            {
                j = i + 1;
                // if numBlock == 000000000
                while ((x.numBlock[j] == 0) && (j != 35))
                {
                    x.numBlock[j] = 999999999;
                    j++;
                }
                x.numBlock[j] -= 1;
            }
        }
        else
        {result.numBlock[i] = x.numBlock[i] - y.numBlock[i];}
    }
    return result;
}

uint1024_t mult_op (uint1024_t x, uint1024_t y)
{
    uint1024_t result;
    uint64_t mult;
    int index;

    int filler = 0;
    memset(&result, filler, sizeof(uint32_t)*35); // allocate memory for num
       blocks

    for (int i = 0; i < 35; i++) // every block in x multiply by block in y
    { // y index
        for (int j = 0; i+j < 35; j++)
        { // x index
            if (x.numBlock[j] && y.numBlock[i])
            { // check blocks for 0s
                // 32bit block x 32 bit block
                mult = ((uint64_t)x.numBlock[j])*y.numBlock[i];
                // resulting array index
                index = i + j;
                while (mult != 0)
                { // check mult != 0
                    result.numBlock[index] += mult % 1000000000; // resulting
                       block += mult div 1000000000
                    mult /= 1000000000; // mult is now 1000000000 times less
                    if (result.numBlock[index] >= 1000000000) // overflow
                    {
                        result.numBlock[index] %= 1000000000; mult += 1;
                    }
                    index++;
                }
            }
        }
    }
    return result;
}
```

3 uint1024_oi.h

```
#include <stdio.h>
#include <locale.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>

void printf_value (uint1024_t x)
{
    int index = 35-1;
    while (index >= 0 && x.numBlock[index] == 0)
    {
        index--; // moving pointer to first numblock that is not 000000000
    }

    if (index < 0) / in case x == 0 (all numBlocks == 000000000)
    {
        printf("0"); return;
    }

    printf("%u", x.numBlock[index]); // throw away leading 0s
    index--;

    while (index >= 0)
    {
        printf("%u", x.numBlock[index]);
        index--;
    }
}

void scanf_value(uint1024_t *x)
{
    char str[310]; // string of 310 symbols
    scanf("%s", str);// scan value as string
    memset(x, 0, sizeof(uint32_t)*35); // allocate memory for num blocks
    int position, index;
    char block[10]; block[9]='\0'; // ending zero (for ATOI function)

    int numLen == strlen(str)-9;
    for (position = numLen, index = 0; position >= 0; position -= 9, index++)
    {
        strncpy(block, str+position, 9); // copy from str starting at position
           to block 9 symbols
        x->numBlock[index] = atoi(block); // put num (9 symbols) in numBlock[i]
    }

    if (position % 9 != 0) // overflow (last block != 0)
    { // put last block in numBlock
        int rest = (position + 9) % 9;
        strncpy(block, str, rest); block[rest] = '\0';
        x->numBlock[index] =  atoi(block);
    }
}
```