МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Обработка текстов. Реализация длинной арифметики на C++.»

Студент гр. 3352 _____ Рябов В.А.

Преподаватель _____ Калмычков В.А.

Санкт-Петербург

2024

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Рябов В.А.

Группа 3352

Тема работы: Создание обучающей системы, предназначенная для школьников 1–2-го классов средней школы, которая позволяет выполнять базовые арифметические операции с контролем хода вычислений и полученных результатов над целыми n-разрядными положительными числами (n < 80).

Исходные данные:
Дан список чисел, список запросов для этих чисел.

Содержание пояснительной записки:
"Исходная формулировка задания", "Математическая постановка задачи", "Описание списков", "Особенности реализации на компьютере", "Контрольные примеры", "Организация диалога с пользователем", "Раздел описания классов", "Формат хранения данных", "Функции программы", "Порядок использования файлов", "Алгоритм решения", "Программа"

Предполагаемый объем пояснительной записки:
Не менее 30 страниц.
Дата выдачи задания: 20.02.2024
Дата сдачи реферата: .06.2024
Дата защиты реферата: .06.2024

Студент _____ Рябов В.А.

Преподаватель _____ Калмычков В.А.

АННОТАЦИЯ

В данной курсовой работе требуется реализовать обучающую систему для школьников 1-2 класса. Система должна демонстрировать базовые операции сложения, вычитания, умножения, деления чисел. Все числа имеют длину не более 80 символов и хранятся в списках в виде линейных связанных списков целых чисел. Методами создания программы стали изученные в ходе семестра возможности языка программирования C++, а именно классы, структуры и списки.

SUMMARY

In this course work, it is required to implement a training system for schoolchildren in grades 1-2. The system must demonstrate basic operations of addition, subtraction, multiplication, and division of numbers. All numbers are no more than 80 characters long and are stored in lists as linear lists of integers. The methods for creating the program were the capabilities of the C++ programming language studied during the semester, namely classes, structures and lists.

СОДЕРЖАНИЕ

Введение.

Целью работы является реализация программы на языке C++ с использованием знаний, полученных в ходе 1 и 2 семестра, обучится использованию связанных списков, а также вложенных связанных списков.

Исходная формулировка задания.

Обучающая система, предназначенная для школьников 1–2-го классов средней школы должна позволять демонстрировать выполнение арифметических операций (сложение, вычитание, умножение и деление) над целыми n-разрядными положительными числами (n < 80) выполнять арифметические операции с контролем хода вычислений и полученных результатов.

Математическая постановка задачи.

В работе необходимо проведение арифметических операций над длинными числами. Так как числа достаточно длинные, то они хранятся в списках char по 5 символов в каждом блоке. Это удобно для считывания и хранения, но не для выполнения операций. Поэтому, числа переводятся в массивы цифр типа int.

Рассмотрим методы сложения и вычитания. Реализация их похожа. Для начала списки будет необходимо отзеркалить, как показано на рисунке:

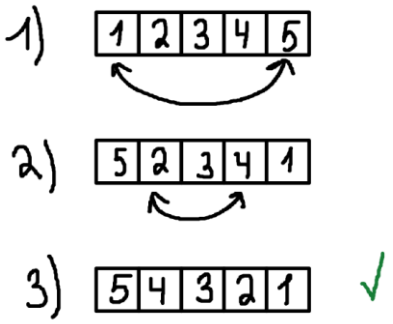


Рис. 1 – Переворачивание списка.

Далее, мы будем складывать числа по разрядам прибавлять остаток, если число + остаток больше 9 то в ответ пишем последнюю цифру если нет то просто число + остаток. Сохраняем остаток.

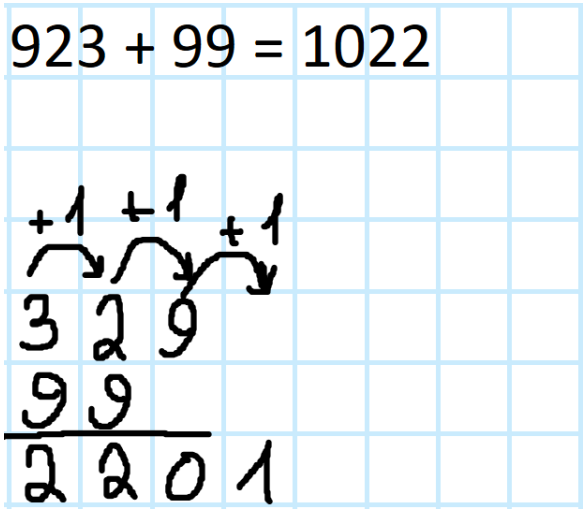


Рис. 2 – Реализация сложения в программе.

Перед вычитанием мы убеждаемся, что вычитаем из большего меньшее. Вычитание реализуется аналогично, только мы будем вместо сложения цифр их вычитать. Если получим отрицательное число, то к результату прибавим 10, а из следующего разряда вычитаемого вычитаем 1.

Теперь рассмотрим умножение. Мы выделим память под результат, он будет равен сумме размеров множителей. Будем умножать цифры второго множителя на 1 добавлять остаток и то, что у нас находится в i+j разряде. Так как результат мы будем сразу складывать, то вывод “лесенкой” у нас не получится, поэтому перед умножением мы выведем лесенку, просто умножив цифру у второго множителя на первый множитель, не забудем сделать отступ.

$$123456 \times 12345 = 1524064320$$

			×	1	2	3	4	5	6
					1	2	3	4	5
				6	1	7	2	8	0
			4	9	3	8	2	4	
		3	7	0	3	6	8		
	2	4	6	9	1	2			
1	2	3	4	5	6				
1	5	2	4	0	6	4	3	2	0

Рис. 3 – Умножение лесенкой.

Рассмотрим деление. Сразу обработаем случаи, когда делитель = 0. Когда делитель больше делимого просто запишем 0. Когда числа равны запишем 1.

Далее, из делимого с начала будем брать по цифре, сохранять ее, затем проверять, больше ли она делителя.

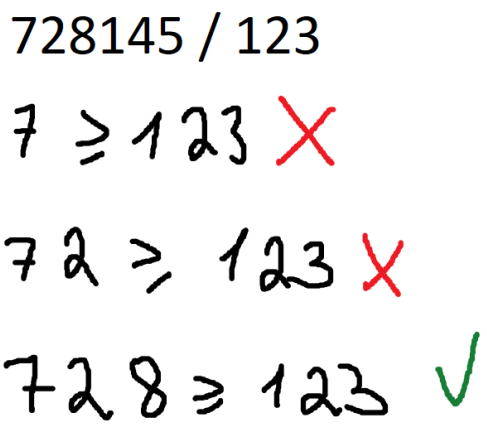


Рис. 4 – Подбор.

Когда число найдено, начинаем умножать делитель на числа от 1 до 9 пока не найдем равное или число большее.

```
Шаг: 1 получен 123 >= 723 - нет
Шаг: 2 получен 246 >= 723 - нет
Шаг: 3 получен 369 >= 723 - нет
Шаг: 4 получен 492 >= 723 - нет
Шаг: 5 получен 615 >= 723 - нет
Шаг: 6 получен 738 >= 723 - да, значит берем меньшее
728 - 615 = 113
Получил промежуточное число: 113
В ответ добавляется цифра 5
```

Рис. 5 – Процесс деления.

И так далее. Не забудем потом, что когда мы сносим сразу две цифры сразу, то в результат нужно добавить 0. Когда сносить будет нечего, мы прекратим деление и получим результат.

```
 728145|123
-615   |5919
 1131
-1107
  244
 -123
  1215
 -1107
```

Рис. 6 – Деление столбиком.

Описание списков.

Для хранения данных используются связанные двунаправленные списки. Устройство списков можно видеть ниже:

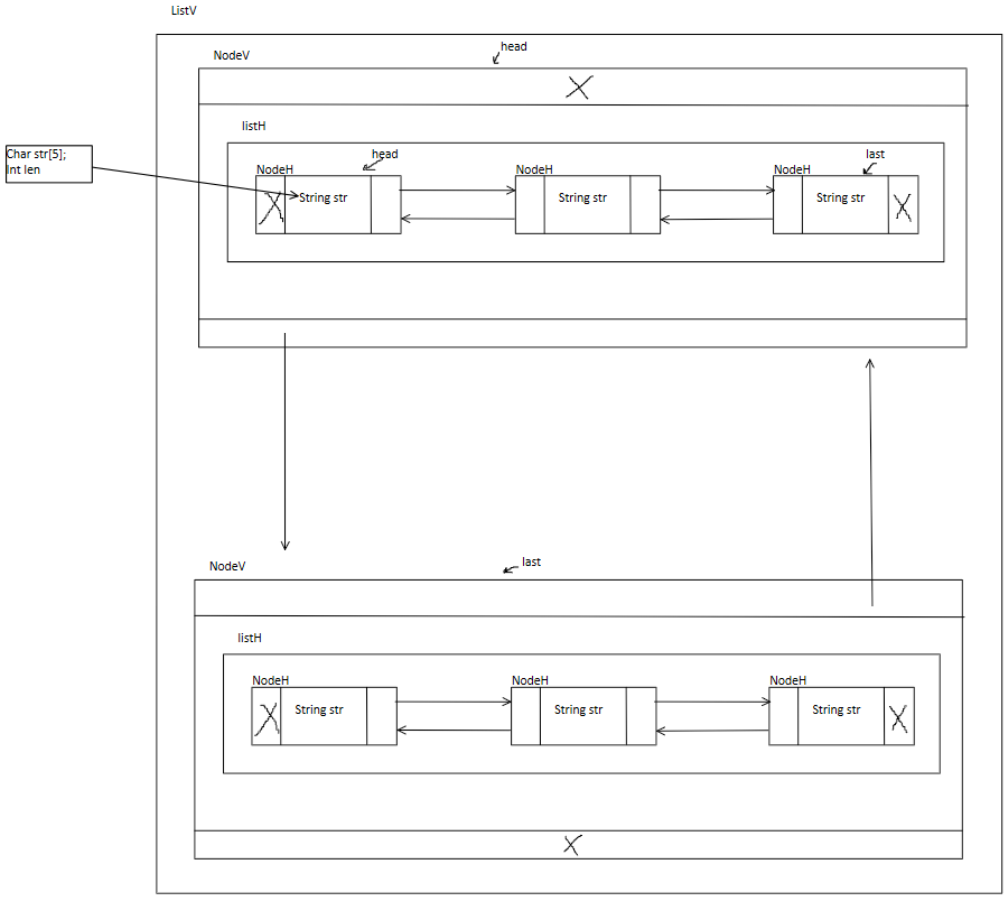


Рис. 7 – Организация списков.

Особенности реализации на компьютере.

В программе используются связанные вертикальные двунаправленные списки, которые, в свою очередь, хранят в себе горизонтальные списки, в которых находятся элементы строк, хранящие по 5 символов. Этот формат удобен для хранения, но неудобен для реализации арифметических операций. Поэтому в программе используются дополнительные методы, которые переводят из линейных списков во временные динамические массивы чисел типа Int. После выполнения операции память освобождается.

Список запросов также хранится в виде линейного списка, что позволяет удобно с ними работать. Для каждого списка имеет перегруженный оператор [], который позволяет удобно обратиться к элементу по индексу, без использования громоздких циклов и указателей.

Контрольные примеры.

Вид файла с запросами:

```
123
12
123
699110
52
123456789
987654321
1234
13
728145
97652851
100000371
200252352525
15612352525
865802619705879147149759275
082957209750928592095 7
```

```
+ 1 2
/ 8 9 10
/ 4 5
/ 1 3
* 10 11 11
/ 10 3 9
* 1 2
- 15 16
```

Пример №1. Сложение больших чисел.

Рассмотрим пример, где складываются весьма большие числа.

```
15. 865802619705879147149759275
16. 0829572097509285920957
```

Выполнив нужную операцию, получим:

```
8. Число под номером 15 + Число под номером 16
```

```
Результат сложения чисел:

       865802619705879147149759275
            0829572097509285920957
       ---------------------------
       865803449277976656435680232
```

Пример №2. Вычитание

Рассмотрим пример с вычитанием этих же самых чисел.

```
15. 865802619705879147149759275
16. 0829572097509285920957
```

Выполнив операцию вычитания, получим:

```
Результат вычитания чисел:

       865802619705879147149759275
            0829572097509285920957
       ---------------------------
       865801790133781637863838318
```

Пример №3. Умножение чисел.

Выполним 5 запрос:

```
5. Число под номером 10 * Число под номером 11
```

Получим:

```
Результат умножения чисел:

              728145
             97652851
        --------------
              728145
             3640725
            5825160
           1456290
          3640725
         4368870
        5097015
       6553305
        --------------
       71105435191395
```

Пример №4. Деление.

Выполним запрос с делением.

```
Результат деления чисел:

        728145|123
        615   |
        -------
        1131
        1107
          244
          123
          1215
          1107
        Ответ: 5919
Результат сохранен на 9 место списка.
```

Пример №5. Деление меньшего на большее.

```
Результат деления чисел:

        52|699110
        Ответ: 0
Результат сохранен в конец списка!
```

Организация диалога с пользователем.

Диалог с пользователем происходит через файл. В файле “Input.txt“ пользователь вводит числа, в “Requests.txt” запрос.

Таблица 1 – Пример организации UI в работе.

Варианты ввода	Для файла чисел: числа количеством символов < 80. Для файла запроса (каждое через пробел): (операция) (номер 1 числа) (номер 2 числа) (номер куда сохранить) Для консоли: на ввод принимаются только числовые значения нужного запроса.
Варианты вывода в файлы	Файл чисел “Input.txt“: 100000371 200252352525 15612352525 865802619705879147149759275 0829572097509285920957 Файл запросов “Requests.txt”: + 1 2 2 / 4 3 * 1 4 - 4 5 1
Консольный вывод	Файл успешно считан. Полученный список: 1. 100000371 2. 200252352525 3. 15612352525 4. 865802619705879147149759275 5. 082957209750928592095 7 Введи нужный номер запроса: 1. Число под номером 1 + Число под номером 2 2. Число под номером 4 / Число под номером 3 3. Число под номером 1 * Число под номером 4

	4. Число под номером 4 - Число под номером 5 0. Назад. Введите номер:
Варианты ответа на вводимый номер.	1. Успешное выполнение операции: Введите номер: 1 Результат сложения чисел: 100000371 200252352525 ---------- 200352352896 Результат сохранен на 2 место списка. Полученный список: 1. 100000371 2. 200352352896 3. 15612352525 4. 865802619705879147149759275 5. 0829572097509285920957 2. Ошибка пользователя при вводе: Введите номер: 1488 Такого запроса нет! 3. Отсутствие номера: 5. Число под номером 20 + Число под номером 31 0. Назад. Введите номер: 5 Номера нужных строк не найдены!

Раздел описания классов.

Название	Входящие переменные	Назначение входящих переменных
class String	char str[N];	Массив символов
	int len = N;	Количество элементов в списке
	int Len();	Возвращает размер элемента.
	char& operator[](int index);	Оператор []. Возвращает ссылку на символ.
	void SetElem(char elem, int index);	Изменить элемент по индексу.
	void SetLen(int newlen);	Установить размер.
	int Znach();	Возвращает число.
	void Clear();	Очистка строки.
	friend std::ostream& operator<<(std::ostream& os, String& string);	Оператор вывода в поток.

Название	Входящие переменные	Назначение входящих переменных
class NodeH	String elem;	Строка
	NodeH* strPrev;	Указатель на предыдущий элемент
	NodeH* strNext;	Указатель на следующий элемент
	NodeH(String& str, NodeH* strPrev=nullptr, NodeH* strNext=nullptr);	Конструктор
	~NodeH();	Деструктор

Название	Входящие переменные	Назначение входящих переменных
class ListH	NodeH* head;	Указатель на начало

	NodeH* cur;	Указатель на текущий элемент
	NodeH* last;	Указатель на последний элемент
	int size = 0;	Размер списка
	ListH();	Пустой конструктор
	ListH(const ListH& other);	Конструктор копирования
	void push_back(String& str);	Добавление в конец списка.
	void copyList(const ListH& lst);	Копировать элементы из другого списка.
	int GetSize();	Получить размер списка.
	String& operator[](int index);	Оператор []
	friend std::ostream& operator<<(std::ostream& os, ListH& listH);	Оператор вывода поток (выводит в поток список)

Название	Входящие переменные	Назначение входящих переменных
class NodeV	ListH elemH;	Горизонтальный список
	NodeV* lstPrev;	Указатель на предыдущий
	NodeV* lstNext;	Указатель на следующий
	NodeV(ListH& elemH, NodeV* lstPrev = nullptr, NodeV* lstNext = nullptr);	Конструктор

Название	Входящие переменные	Назначение входящих переменных
class ListV	NodeV* head;	Указатель на начало
	NodeV* cur;	Указатель на текущий элемент
	NodeV* last;	Указатель на последний элемент
	int size;	Размер списка
	ListV();	Конструктор

	~ListV();	Деструктор
	void push_back(ListH& list);	Добавить в конец списка.
	ListH& operator[](int index);	Оператор [] (получить горизонтальный список по индексу)
	void pop_back();	Удалить последний элемент
	void pop(int n);	Удалить элемент по индексу
	void OutList(std::ostream& os);	Вывести список
	void push(ListH& list, int pos);	Сохранить элемент в определенное место.

Формат хранения данных.

В таблице не рассматриваются переменные счетчиков, а также временные переменные, которые не выходят за области видимости своих локальных функций.

Таблица 2 – Переменные в программе.

Тип переменной	Название	Значение
ifstream	file	Файл с числами “Input.txt”.
	request	Файл с запросами “Request.txt”
ofstream	ofile	Файл для вывода.
	log	Файл для логирования.
ListV	list	Список чисел
	requests	Список запросов
int	n	Число введенное пользователем

Средства обеспечения ввода/вывода.

Библиотека	Команды
iostream	cout, cin
fstream	ifstream, ofstream, .eof(), .open()
locale	setloacle
iomanip	setw, setfill

Функции программы.

Таблица 4 – Функции, использованные в программе.

Имя функции	Назначение	Параметры				Возвращаемое значение	Внешние эффекты.
		Входные	Выходные	Модифицированные	Транзитные		
ReverseInt	Переворачивание строки	int* number, int size	-	int* number		void	
RemoveZeroes	Удаление ведущих нулей	int* number, int& size		int* number, int& size		void	
Oequal	Операция сравнения на равенство	int* number1, const int sizel1, int* number2, const int sizel2	bool			bool	
Oless	Операция сравнения на меньше	int* number1, const int sizel1, int* number2, const int sizel2	bool			bool	
Orless	Операция сравнения на меньше или равно	int* number1, const int sizel1, int* number2, const int sizel2	bool			bool	
Obig	Операция сравнения на больше или равно	int* number1, const int sizel1, int* number2, const int sizel2	bool			bool	
Addition	Сумма чисел	int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr		number1, number2, result, sizelr	result, sizelr	void	
Sub	Вычитание	int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr		number1, number2, result, sizelr	result, sizelr	void	
Multiply	Умножение	int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr		number1, number2, result, sizelr	result, sizelr	void	
Out	Метод для вывода числа	int* number, int size				char*	
Subtract	Функция вычитания number2 из	int* number1, int size1, int* number2, int size2		number1, number2		void	

	number1 (для int*)						
MultiplyByNumber	Умножение массива на число	int* number, int& size, int n, int* result		result		void	
Division	Деление	int* number1, const int sizel1, int* number2, const int sizel2, ListH& result_list, int& sizelr		number1, number2, result, sizelr	result, sizelr	void	
CreateArray	Превращает списки в массивы чисел	ListH& l1, ListH& l2, int* number1, int sizel1, int* number2, int sizel2		number2, number1		void	Заполнение массивов.
MakeList	Создать из Int список	ListH& list, int* number, int size				void	
DoRequest	Выполнение запроса	ListV& list, ListH& request				void	
ReadRequest	Читает файл с запросом	ListV& requests				void	
RequestHandler	Основной метод с запросами из файла	ListV& list				void	
main	Основной метод программы			ListH list		int	Открытие файлов. Выполнение основной функции в программе.

Порядок использования пользовательских файлов.

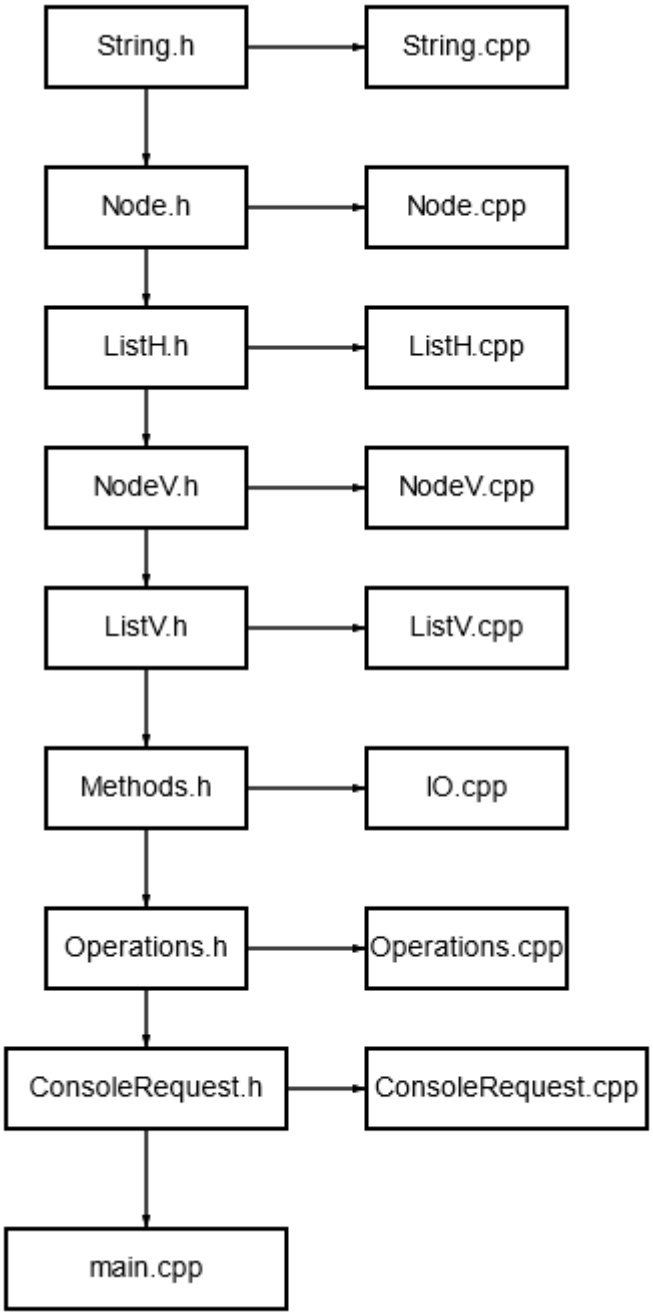
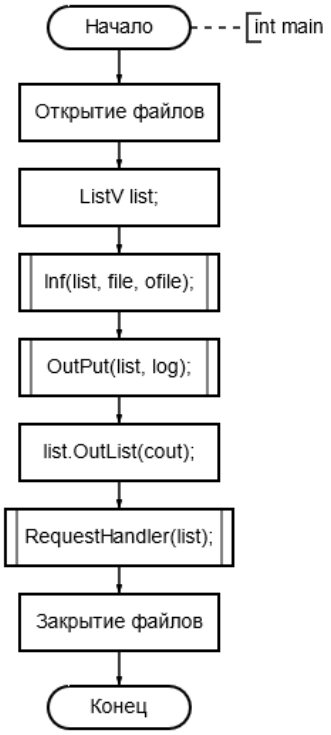


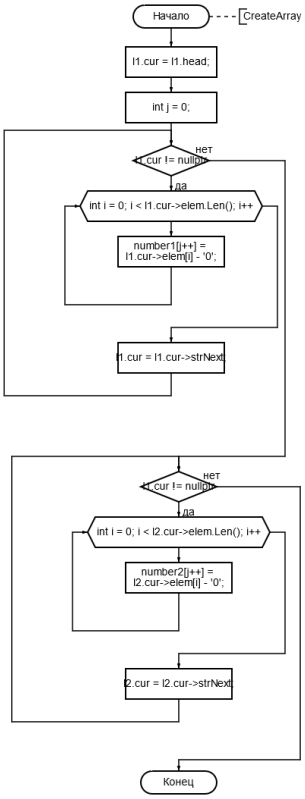
Рис. 8 – Организация файлов.

Алгоритм решения.

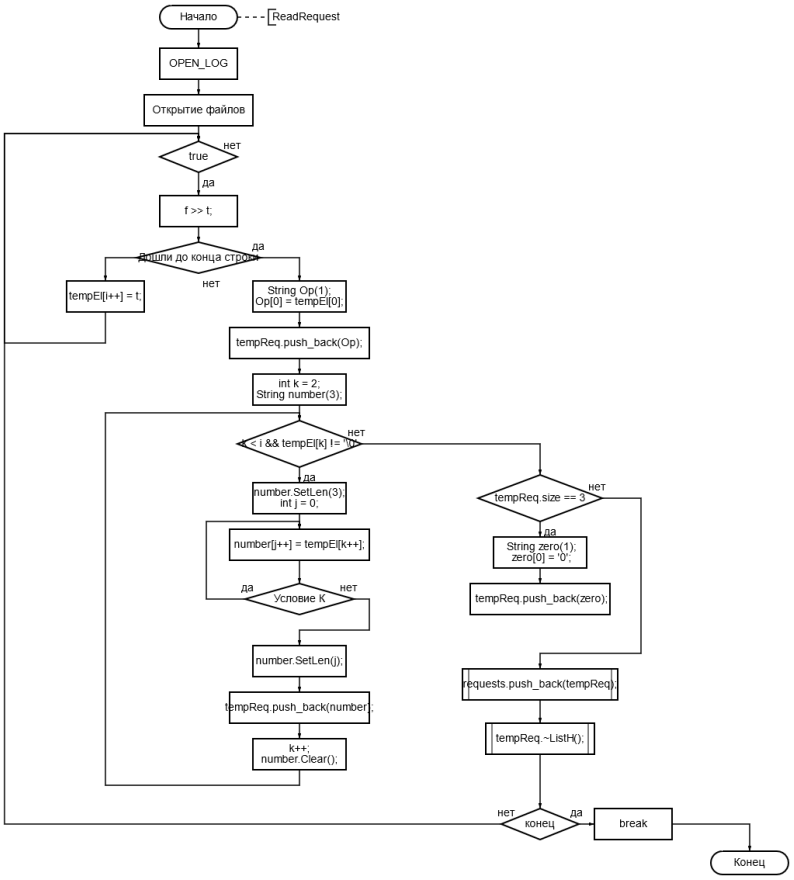
int main:



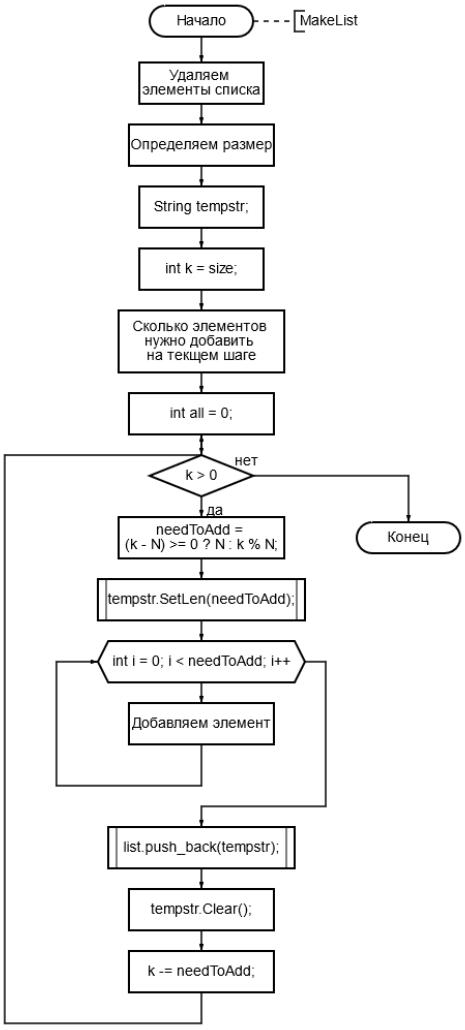
CreateArray:



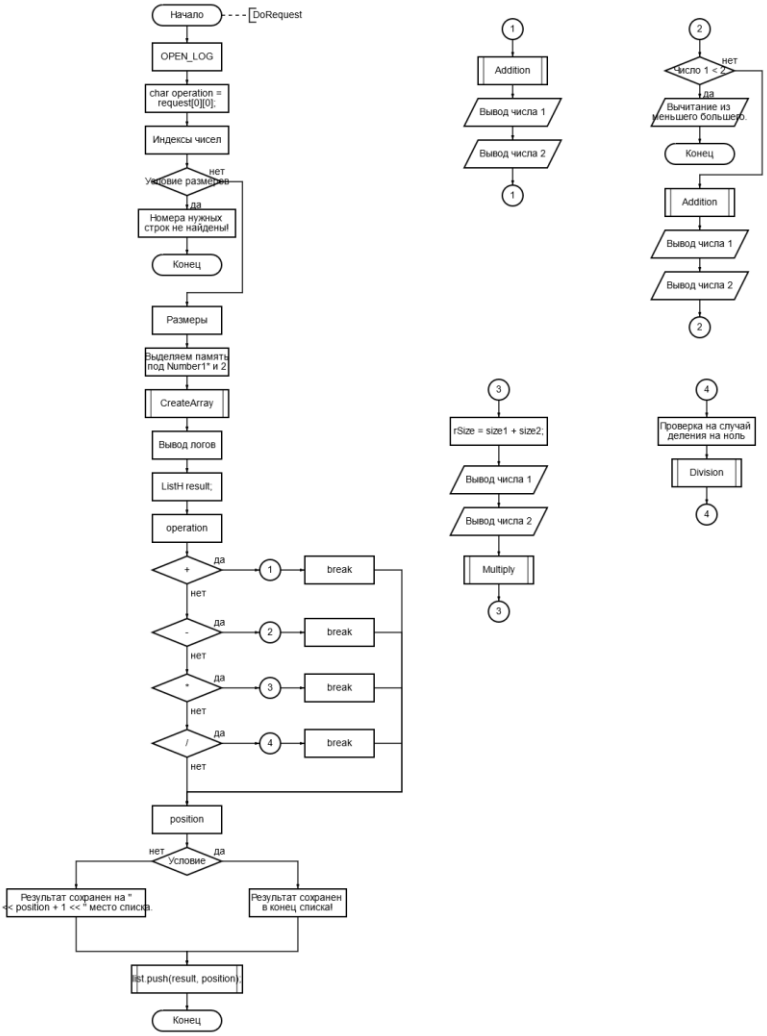
ReadRequest:



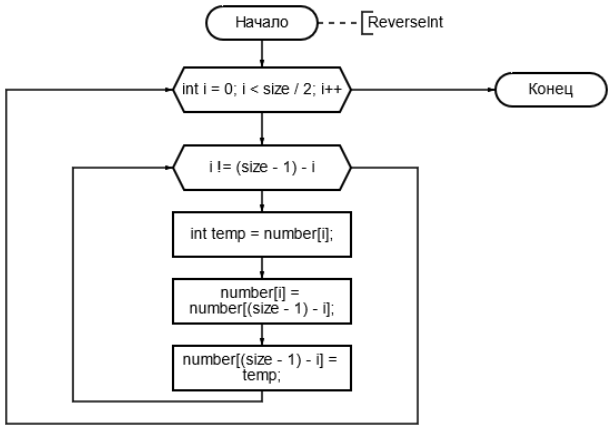
MakeList:



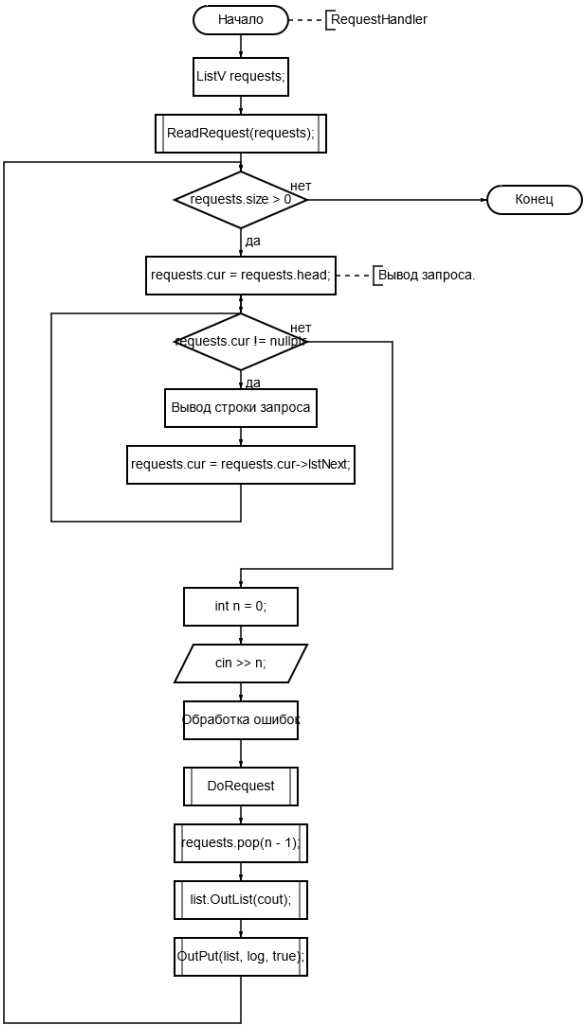
DoRequest:



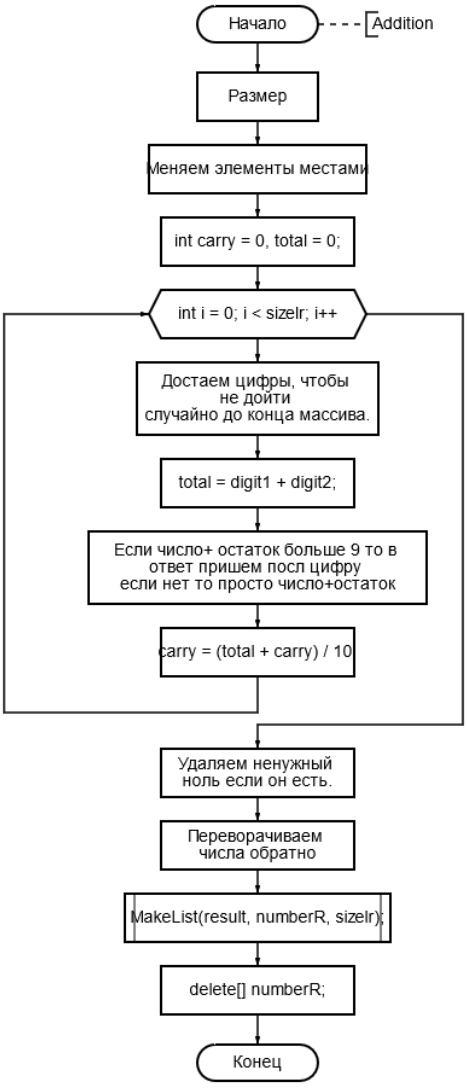
ReverseInt:



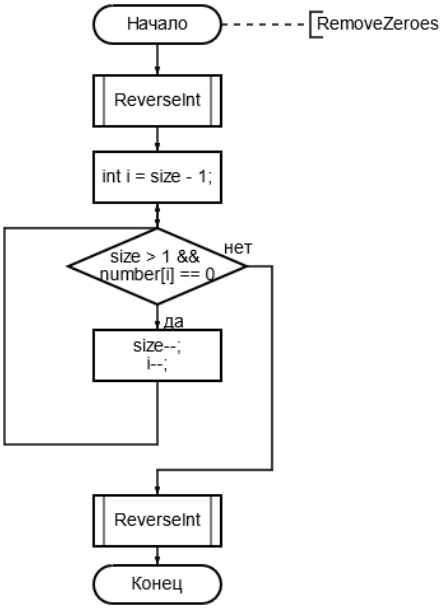
RequestHandler:



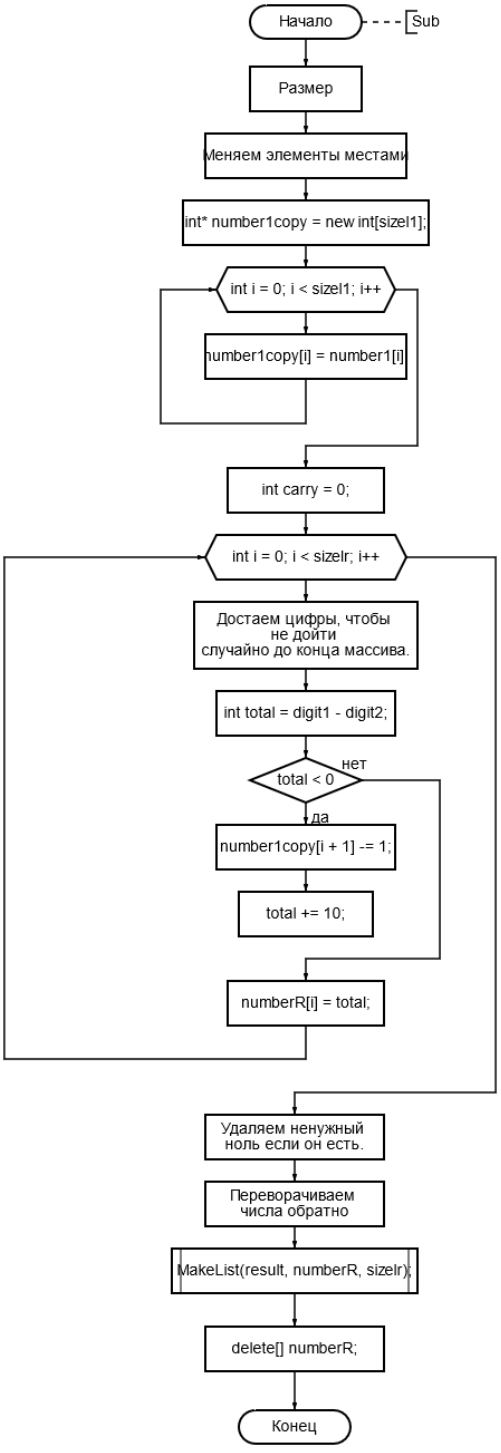
Addition:



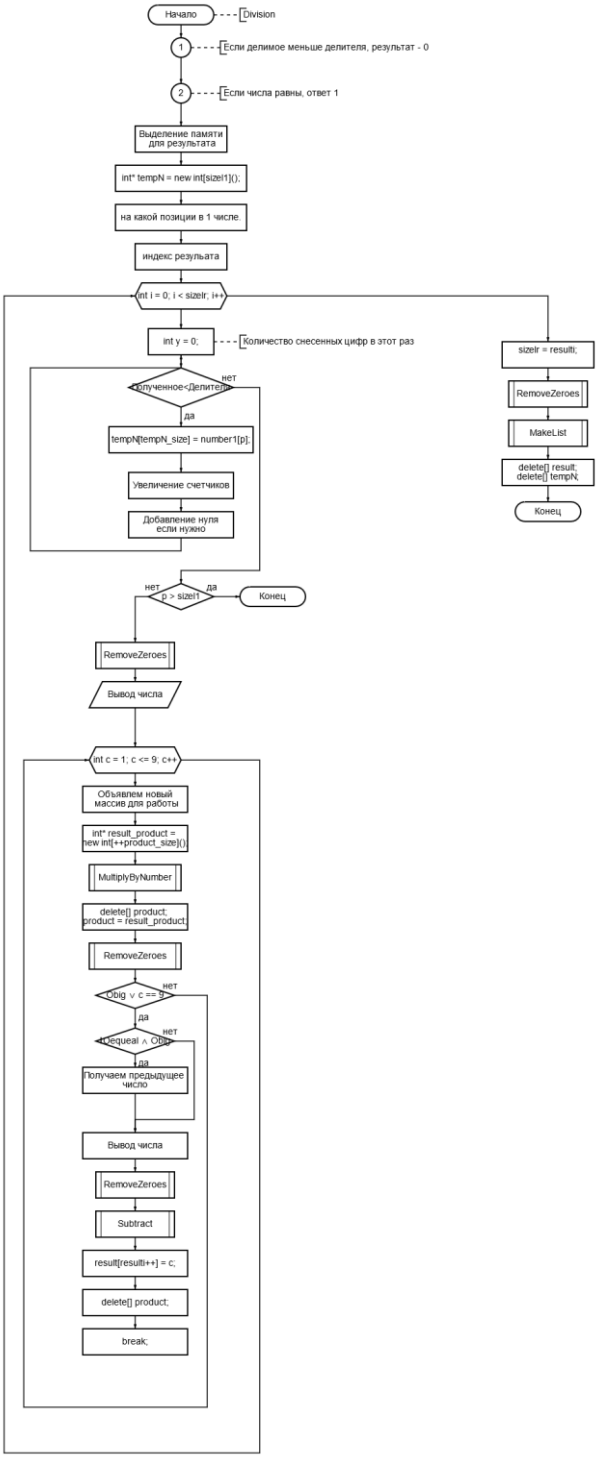
ReverseInt:



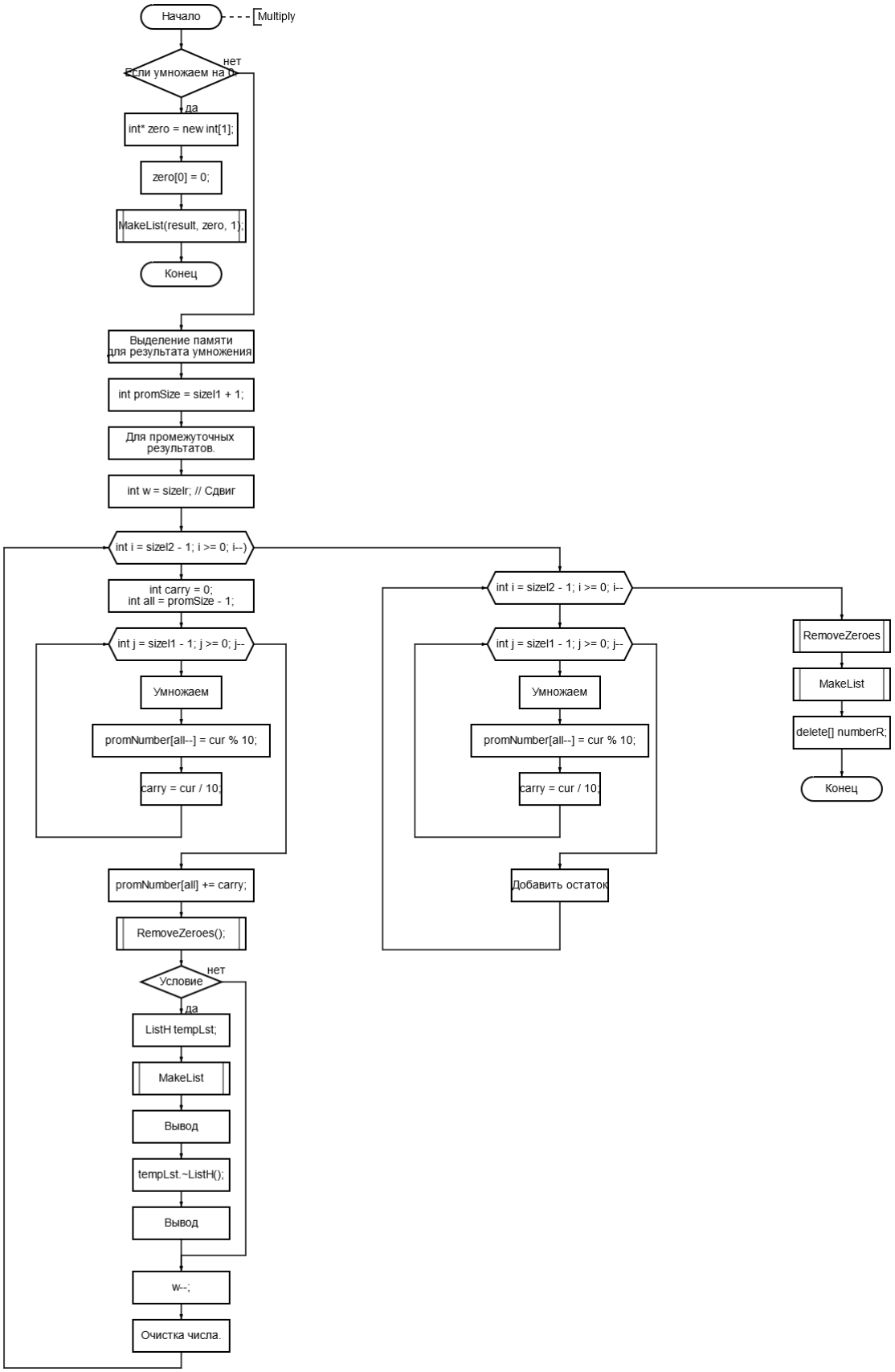
Sub:



Division:



Multiply:


Начало
Multiply
Если умножаем на 0
нет
да
int* zero = new int[1];
zero[0] = 0;
MakeList(result, zero, 1);
Конец
Выделение памяти для результата умножения
int promSize = sizel1 + 1;
Для промежуточных результатов.
int w = sizeIr; // Сдвиг
int i = sizel2 - 1; i >= 0; i--)
int carry = 0;
int all = promSize - 1;
int j = sizel1 - 1; j >= 0; j--
Умножаем
promNumber[all--] = cur % 10;
carry = cur / 10;
promNumber[all] += carry;
RemoveZeroes();
Условие
нет
да
ListH tempLst;
MakeList
Вывод
tempLst.~ListH();
Вывод
w--;
Очистка числа.
int i = sizel2 - 1; i >= 0; i--
int j = sizel1 - 1; j >= 0; j--
Умножаем
promNumber[all--] = cur % 10;
carry = cur / 10;
Добавить остаток
RemoveZeroes
MakeList
delete[] numberR;
Конец

Программа.

ConsoleRequest.cpp

```
#include "ConsoleRequest.h"
using namespace std;
// Превращает списки в массивы чисел.
void CreateArray(ListH& l1, ListH& l2, int* number1, int sizel1, int*
number2, int sizel2)
{
	// Для 1 списка.

	l1.cur = l1.head;
	int j = 0; // Переменная для пробега по массиву int.
	while (l1.cur != nullptr)
	{
		for (int i = 0; i < l1.cur->elem.Len(); i++)
		{
			number1[j++] = l1.cur->elem[i] - '0';
		}
		l1.cur = l1.cur->strNext;
	}

	// Тоже для второго списка.
	//
	// Для 2 списка.

	l2.cur = l2.head;
	j = 0; // Переменная для пробега по массиву int.
	while (l2.cur != nullptr)
	{
		for (int i = 0; i < l2.cur->elem.Len(); i++)
		{
			number2[j++] = l2.cur->elem[i] - '0';
		}
		l2.cur = l2.cur->strNext;
	}
}

// Создать из Int список
void MakeList(ListH& list, int* number, int size)
{
	// Удаляем жлементы списка, если он уже заполнен.
	list.~ListH();

	//int j = 0; // Счетчик

	int listSize = size / N + (size % N != 0);

	String tempstr;

	int k = size; // сколько элементов осталось добавить.

// Вспомогательные методы для деления.

// Метод для вывода числа.
std::string Out(int* number, int size)
{
	std::ostringstream oss;
	for (int i = 0; i < size; i++) {
		oss << number[i];
	}
	return oss.str();
}

// Функция вычитания number2 из number1 (number1 >= number2)
void Subtract(int* number1, int size1, int* number2, int size2)
{
	OPEN_LOG
		log << Out(number1, size1) << " - " <<
Out(number2, size2) << " = ";
	for (int i = 0; i < size2; i++) {
		number1[size1 - size2 + i] -= number2[i];
		if (number1[size1 - size2 + i] < 0) {
			number1[size1 - size2 + i] += 10;
			number1[size1 - size2 + i - 1] -= 1;
		}
	}
	// Обработка заемов
	for (int i = size1 - size2 - 1; i >= 0; i--) {
		if (number1[i] < 0) {
			number1[i] += 10;
			number1[i - 1] -= 1;
		}
	}
	log << Out(number1, size1) << endl;
	CLOSE_LOG
}

// Умножение массива на число.
void MultiplyByNumber(int* number, int& size, int n, int* result)
{

	int result_size = size; // Результат может быть на 1 разряд
больше исходного числа

	size--;

	int carry = 0;
	for (int i = size - 1; i >= 0; i--) {
		int product = number[i] * n + carry;
		result[i + 1] = product % 10;
		carry = product / 10;
	}
	result[0] = carry;

	RemoveZeroes(result, result_size);

	size = result_size;

}

// Деление
void Division(int* number1, const int sizel1, int* number2, const int
sizel2, ListH& result_list, int& sizelr)
{
	OPEN_LOG

		// Если делимое меньше делителя, результат -
0
		if (Oless(number1, sizel1, number2, sizel2))
		{
			int* zero = new int[1];
			zero[0] = 0;
			MakeList(result_list, zero, 1);
			log << "Результат: " << 0 << endl;
```

```
	int needToAdd = (k - N) >= 0 ? N : k % N; // Сколько элементов нужно добавить на текщем шаге.
	int all = 0;
	while (k > 0)
	{
		needToAdd = (k - N) >= 0 ? N : k % N;

		tempstr.SetLen(needToAdd);


		for (int i = 0; i < needToAdd; i++)
		{
			tempstr[i] = number[all++] + '0';
		}

		list.push_back(tempstr);
		tempstr.Clear();

		k -= needToAdd;
	}
}

// Выполнение запроса.
void DoRequest(ListV& list, ListH& request)
{
	OPEN_LOG



	char operation = request[0][0];

	int i1 = request[1].Znach() - 1, i2 = request[2].Znach() - 1; // Индексы чисел



	if (i1 >= list.size || i2 >= list.size)
	{
		cout << RED << "Номера нужных строк не найдены!\n" << RESET;
		return;
	}

	int size1 = list[i1].GetSize(), size2 = list[i2].GetSize(); // Размеры

	int* number1 = new int[size1];
	int* number2 = new int[size2];


	CreateArray(list[i1], list[i2], number1, size1, number2, size2);
		cout << '\t' << "Ответ: " << 0 << endl;
		delete[]zero;
		return;
	}

// Если числа равны, ответ 1
if (Oequal(number1, sizel1, number2, sizel2))
{
	int* one = new int[1];
	one[0] = 1;
	MakeList(result_list, one, 1);
	log << "Результат: " << 1;
	cout << '\t' << "Ответ: " << 1 << endl;
	delete[]one;
	return;
}


// Выделение памяти для результата
sizelr = sizel1 - sizel2 + 1;
int* result = new int[sizelr]();


int* tempN = new int[sizel1]();
int tempN_size = 0;

int p = 0; // на какой позиции в 1 числе.
int resulti = 0; // индекс резульата
for (int i = 0; i < sizelr; i++)
{
	// Формирую число, на которое делится.
	if (p >= sizel1)
		break;

	// Не забываю, что если сношу две цифры сразу, то ответ нолик
	int y = 0; // Количество снесенных цифр в этот раз.
	while (Orless(tempN, tempN_size, number2, sizel2))
	{
		log << Out(tempN, tempN_size) << "<=" << Out(number2, sizel2) << endl;
		if (p > sizel1)
			break;

		tempN[tempN_size] = number1[p];
		p++;
		tempN_size++;
		y++;

		if (y > 1 && resulti)
		{
			result[resulti++] = 0;
		}

	}
	if (p > sizel1)
		break;

	log << Out(tempN, tempN_size) << ">=" << Out(number2, sizel2) << endl;


	RemoveZeroes(tempN, tempN_size);


	if (i != 0)
		cout << '\t' << setw(p) << Out(tempN, tempN_size) << endl;
	// Начинаем вычитать нужно сделть копию или двоичный поиск.
```

```
	log << "Выполняется операция: " << operation << " над числами " << list[i1] << " и " << list[i2] << endl;

	log << "Вид чисел в программе: \n";

	OutStrH(list[i1], log, true);
	log << endl;
	OutStrH(list[i2], log, true);
	log << endl;

	log << "Числа переведены в массивы int рамерами" << size1 << " и " << size2 << " соответственно\n";


	int rSize = 0;

	ListH result;

	switch (operation)
	{
	case '+':

		Addition(number1, size1, number2, size2, result, rSize);


		cout << GREEN << "Результат сложения чисел: \n\n" << RESET;
		cout << '\t' << setw(rSize) << list[i1] << endl
			<< '\t' << setw(rSize) << list[i2] << endl
			<< '\t' << setfill('-') << setw(rSize) << "" << endl
			<< '\t' << result << "\n\n";



		log  << "Результат сложения чисел: \n\n";
		log << '\t' << setw(rSize) << list[i1] << endl
			<< '\t' << setw(rSize) << list[i2] << endl
			<< '\t' << setfill('-') << setw(rSize) << "" << endl
			<< '\t' << result << "\n\n";


		break;


	case '-':


		if (Oless(number1, size1, number2, size2))
		{
```

```
			for (int c = 1; c <= 9; c++)
			{
				// Объявлем новый массив для работы
				int product_size = sizel2;
				int* product = new int[product_size]();
				for (int i = 0; i < sizel2; i++)
				{
					product[i] = number2[i];
				}


				int* result_product = new int[++product_size]();
				MultiplyByNumber(product, product_size, c, result_product);
				delete[] product; product = result_product;

				log << "Шаг: " << c << " получен " << Out(product, product_size) << endl;

				RemoveZeroes(product, product_size);

				if (Obig(product, product_size, tempN, tempN_size) || c == 9)
				{
					if (!Oequal(product, product_size, tempN, tempN_size) && Obig(product, product_size, tempN, tempN_size))
					{
						// Нам нужно уже предыдущее.
						c--;


						product_size = sizel2;
						delete[] product;
						product = new int[product_size]();
						for (int i = 0; i < sizel2; i++)
						{
	product[i] = number2[i];
						}


	RemoveZeroes(tempN, tempN_size);

						result_product = new int[++product_size]();
	MultiplyByNumber(product, product_size, c, result_product);
						delete[] product; product = result_product;
	RemoveZeroes(product, product_size);
					}

					cout << '\t' << setw(p) << Out(product, product_size);
					if (i == 0) cout << setw(sizel1 - p + 1) << '|';
					cout << endl;
					if (i == 0)
					{
						cout << '\t' << setw(sizel1 + 1) << setfill('-') << "" << endl;
						cout << setfill(' ') << "";
					}
```

```
                cout << RED << "Нельзя вычитать из меньшего большее!!\n" << RESET;
                log << "Вычитание из меньшего большего. Отмена\n";
                return;
            }
            Sub(number1, size1, number2, size2, result, rSize);
            rSize = max(size1, size2);

            cout << GREEN << "Результат вычитания чисел: \n\n" << RESET;
            cout << '\t' << setw(rSize) << list[i1] << endl
                << '\t' << setw(rSize) << list[i2] << endl
                << '\t' << setfill('-') << setw(rSize) << "" << endl
                << '\t' << setfill(' ') << setw(rSize) << result << "\n\n";

            log << "Результат вычитания чисел: \n\n";
            log << '\t' << setw(rSize) << list[i1] << endl
                << '\t' << setw(rSize) << list[i2] << endl
                << '\t' << setfill('-') << setw(rSize) << "" << endl
                << '\t' << setfill(' ') << setw(rSize) << result << "\n\n";

            break;

        case '*':
            rSize = size1 + size2;

            cout << GREEN << "Результат умножения чисел: \n\n" << RESET;
            cout << '\t' << setw(rSize) << list[i1] << endl
                << '\t' << setw(rSize) << list[i2] << endl
                << '\t' << setfill('-') << setw(rSize) << "" << setfill(' ') << endl;

            log << "Результат умножения чисел: \n\n";
            log << '\t' << setw(rSize) << list[i1] << endl
                << '\t' << setw(rSize) << list[i2] << endl
                << '\t' << setfill('-') << setw(rSize) << "" << setfill(' ') << endl;

                Subtract(tempN, tempN_size, product, product_size);

                RemoveZeroes(tempN, tempN_size);

                log << "Получил промежуточное число: ";
                log << setw(p) << Out(tempN, tempN_size) << endl;
                log << "В ответ добавляется цифра " << c << endl;

                result[resulti++] = c;
                delete[] product;
                break;
            }
        }
    }

    log << "Результат: " << Out(result, resulti);

    sizelr = resulti;

    cout << '\t' << "Ответ: " << Out(result, resulti) << endl;

    RemoveZeroes(result, sizelr);

    MakeList(result_list, result, sizelr);

    delete[] result;
    delete[] tempN;

    CLOSE_LOG
}
```

Operations.h

```
#pragma once
#include "Methods.h"

// Переворачивание строки
void ReverseInt(int* number, int size);

// Удаление ведущих нулей.
void RemoveZeroes(int* number, int& size);

/* Операции сравнения */

// ==
bool Oequal(int* number1, const int sizel1, int* number2, const int sizel2);

// <
bool Oless(int* number1, const int sizel1, int* number2, const int sizel2);

// <=
bool Orless(int* number1, const int sizel1, int* number2, const int sizel2);

// >=
bool Obig(int* number1, const int sizel1, int* number2, const int sizel2);

/* Арифметика */

// Сумма чисел
void Addition(int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr);
```

```
                Multiply(number1, size1, number2, size2, result, rSize);

                rSize = size1 + size2;

                cout << '\t' << setfill('-') << setw(rSize) << "" << endl << setfill(' ')
                        << '\t' << setw(rSize) << result << "\n\n";

                log << '\t' << setfill('-') << setw(rSize) << "" << endl << setfill(' ')
                        << '\t' << setw(rSize) << result << "\n\n";

                break;

        case '/':

                // Проверка на случай деления на ноль
                if (size2 == 1 && number2[0] == 0)
                {
                        cout << RED << "Нельзя вычитать из меньшего большее!!\n" << RESET;
                        log << "Вычитание из меньшего большее. Отмена.\n";
                        return;
                }

                cout << GREEN << "Результат деления чисел: \n\n" << RESET;
                cout << '\t' << list[i1] << "|" << list[i2] << endl;

                log << "Результат деления чисел: \n\n";
                log << '\t' << list[i1] << "|" << list[i2] << endl;

                Division(number1, size1, number2, size2, result, rSize);

                break;
        }

        int position = request[3].Znach() - 1;
        cout << YELLOW;
        if (position >= list.size || position <= 0)
        {
                cout << "Результат сохранен в конец списка!\n";
                log << "Результат сохранен в конец списка!\n";
        }
```

```
// Вычитание
void Sub(int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr);

// Умножение
void Multiply(int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr);

// Вспомогательные методы для деления.

// Метод для вывода числа.
std::string Out(int* number, int size);

// Функция вычитания number2 из number1 (number1 >= number2)
void Subtract(int* number1, int size1, int* number2, int size2);

// Умножение массива на число.
void MultiplyByNumber(int* number, int& size, int n, int* result);

// Деление
void Division(int* number1, const int sizel1, int* number2, const int sizel2, ListH& result_list, int& sizelr);
```

IO.cpp

```
#include"Methods.h"
#include<fstream>
#include<ostream>



bool ReadStr(ListH& lst, std::ifstream& f, int& bytes);

// Считывание файла
void Inf(ListV& two_list, std::ifstream& f, std::ofstream& output)
{
        ListH temp_lst;
        f.clear(); f.unsetf(std::ios::skipws);
        int bytes = 0; // Количесвто считаных символов.
        while (!f.eof())
        {
                bool result = ReadStr(temp_lst, f, bytes); // Читаем строку.
                if (result == 0)
                {
                        output << "Ошибка при чтении файла. Конец программы."; return;
                }
                two_list.push_back(temp_lst);

                if (!f.eof())
                        temp_lst.~ListH(); // Обнулить временный список.

        }
}

// Чтение строки
bool ReadStr(ListH& lst, std::ifstream& f, int& bytes)
{
        if (f.eof()) return 0;

        String tempStr;
        char t{};
        int i = 0;
        unsigned ps = 0;
        while (true)
        {
                f >> t; bytes++;
                if (i >= N || t == '\n' || f.eof()) // Put String in ListH
                {
                        tempStr.SetLen(i);
                        lst.push_back(tempStr);
```

```
	else
	{
		cout << "Результат сохранен на " << BLUE << position + 1 << YELLOW << " место списка.\n";
		log << "Результат сохранен на " << position + 1 << " место списка.\n";
	}
	cout << RESET;
	list.push(result, position);

	cout << setfill(' ') << "";

	CLOSE_LOG
}

// Читает файл с запросом.
void ReadRequest(ListV& requests)
{
	OPEN_LOG
	ifstream f("Request.txt"); f.unsetf(ios::skipws);
	if (f.eof()) return;

	ListH tempReq;

	int i = 0;
	char t = 0;

	char tempEl[13]{};

	log << "Начиниаем считывать файл Request.txt\n";
	while (true)
	{
		f >> t;

		//Дошли до конца строки
		if (t == '\n' || f.eof() || i >= 13)
		{
			log << "Дошли до конца строки\n";
			String Op(1);
			Op[0] = tempEl[0];

			log << "Сохраняем оперцаию - это " << tempEl[0] << "\n";
			tempReq.push_back(Op);

			int k = 2;
			String number(3);

			while (k < i && tempEl[k] != '\0')
			{
				number.SetLen(3);
				int j = 0;
				tempStr.Clear();

				i = 0;

				if (f.eof() || t == '\n')
					return 1;

				f.seekg(--bytes);
			}
			else
			{
				tempStr[i++] = t;
			}
		}
	}

void OutStrH(ListH& lst, std::ostream& ofile, bool isProgram);

// Вывод всего списка
void OutPut(ListV& two_list, std::ostream& ofile, bool isProgram)
{
	two_list.cur = two_list.head;
	while (two_list.cur != nullptr)
	{
		OutStrH(two_list.cur->elemH, ofile, isProgram);
		ofile << std::endl;

		two_list.cur = two_list.cur->lstNext;

		if (two_list.cur != nullptr && isProgram)
			ofile << "\n |\n\\|/\n" << std::endl;
		else
			ofile << std::endl;
	}
}

// Вывод строки списка
void OutStrH(ListH& lst, std::ostream& ofile, bool isProgram)
{
	lst.cur = lst.head;
	int s = 0;
	while (lst.cur != nullptr)
	{
		// Вывод элемента String
		s = lst.cur->elem.Len();
		for (int i = 0; i < s; i++)
		{
			ofile << lst.cur->elem[i];
		}

		lst.cur = lst.cur->strNext;

		if (lst.cur != nullptr && isProgram)
			ofile << " --> ";
	}

}
```

ListH.cpp

```
#include "ListH.h"

ListH::ListH()
{
	head = cur = last = nullptr;
	size = 0;
}

ListH::ListH(const ListH& other)
{
	this->copyList(other);
}

void ListH::push_back(String& str)
{
	if (head == nullptr)
	{
		head = new NodeH(str);
		last = head;
```

```
                                        do
                                        {
                                                number[j++] = tempEl[k++];
                                        } while (tempEl[k] != ' ' && j < 3 && tempEl[k] != '\0');
                                        number.SetLen(j);
                                        log << "Сохраняем номер в список временной строки запроса.\n";
        tempReq.push_back(number);
                                        k++;
                                        number.Clear();
                                }
                                if (tempReq.size == 3)
                                {
                                        String zero(1);
                                        zero[0] = '0';
                                        tempReq.push_back(zero);
                                }
                                requests.push_back(tempReq);
                                log << "Сохраняем запрос.\n";
                                tempReq.~ListH();
                                i = 0;
                                for (int l = 0; l < 13; l++) tempEl[l] = '\0';
                                if (f.eof())
                                        break;
                        }
                        else
                        {
                                tempEl[i++] = t;
                        }
        }
        log << "Обработка запроса окончена. Полученный список с запросами в программе :" << endl;
        OutPut(requests, log, true);
                size++;
        }
        else
        {
                NodeH* elem = new NodeH(str, last);
                last->strNext = elem;
                last = elem;
                size++;
        }
}

void ListH::copyList(const ListH& lst)
{
        this->~ListH();
        if (lst.size != 0)
        {
                NodeH* tmp = lst.cur;
                tmp = lst.head;
                while (tmp!=nullptr)
                {
                        this->push_back(tmp->elem);
                        tmp = tmp->strNext;
                }
        }
}

int ListH::GetSize()
{
        cur = head;
        int l = 0;
        while (cur != nullptr)
        {
                l += cur->elem.Len();
                cur = cur->strNext;
        }
        return l;
}

String& ListH::operator[](int index)
{
        if (index > size)
        {
                throw std::out_of_range("Index Out Of Range");
        }

        cur = head;
        for (int i = 0; i < index; i++)
        {
                cur = cur->strNext;
        }

        return cur->elem;
}

void ListH::pop(int n) // Удаление элемента по номеру
{
        cur = head;
        // Доходим до нужного элемента
        for (int i = 0; i < n - 1; i++)
        {
                cur = cur->strNext;
        }

        // Переадресацию указателей
        if (cur != head)
                cur->strPrev->strNext = cur->strNext;
        else
                head = cur->strNext;
        if (cur != last)
                cur->strNext->strPrev = cur->strPrev;
        else
```

Operations.cpp

```
#include "Operations.h"
#include "ConsoleRequest.h"
#include <sstream>


using namespace std;

// Переворачивание строки
void ReverseInt(int* number, int size)
{
	for (int i = 0; i < size / 2; i++)
	{
		if (i != (size - 1) - i)
		{
			int temp = number[i];
			number[i] = number[(size - 1) - i];
			number[(size - 1) - i] = temp;
		}

	}
}

// Удаление ведущих нулей.
void RemoveZeroes(int* number, int& size)
{
	ReverseInt(number, size);

	int i = size - 1;
	while (size > 1 && number[i] == 0) {

		size--;
		i--;
	}

	ReverseInt(number, size);
}

/* Операции сравнения */

// ==
bool Oequal(int* number1, const int sizel1, int* number2, const int
sizel2)
{
	if (sizel1 != sizel2)
		return false;
	else
	{
		for (int i = 0; i < sizel1; i++)
		{
			if (number1[i] != number2[i])
				return false;
		}
		return true;
	}
}

// <
bool Oless(int* number1, const int sizel1, int* number2, const int
sizel2)
{
	if (Oequal(number1, sizel1, number2, sizel2))
		return false;
	else
	{
		if (sizel1 != sizel2)
			return sizel1 < sizel2;
		else
		{
			for (int i = 0; i < sizel1; i++)
			{
				if (number1[i] !=
number2[i])
					return
number1[i] < number2[i];
			}
```

```
		last = cur->strPrev;


	delete cur;
	size--;

}
ListH::~ListH()
{
	if (size != 0)
	{
		cur = head;
		while (cur!=nullptr)
		{
			NodeH* tmp = cur->strNext;
			delete cur;
			cur = tmp;
		}
	}
	head = cur = last = nullptr;
	size = 0;
}

std::ostream& operator<<(std::ostream& os, ListH& listH)
{

	// Если плохо выводить будет, то метод перепишу с
использованием дополнительной памяти.
	int fullLen = listH.GetSize();
	std::unique_ptr<char[]> fullstr (
std::make_unique<char[]>(fullLen+1) );

	listH.cur = listH.head;
	int j = 0;
	while (listH.cur != nullptr)
	{
		for (int i = 0; i < listH.cur->elem.Len(); i++)
		{
			fullstr[j++] = listH.cur->elem[i];
		}


		listH.cur = listH.cur->strNext;
	}
	fullstr[fullLen] = '\0';

	os << fullstr;

	return os;


	/*
	Old
	listH.cur = listH.head;
	while (listH.cur != nullptr)
	{
		os << listH.cur->elem;
		listH.cur = listH.cur->strNext;
	}

	return os;
	*/
}
```

ListH.h

```
#pragma once
#include "NodeH.h"
class ListH
{
public:
	NodeH* head;
	NodeH* cur;
	NodeH* last;
	int size = 0;


	ListH();
```

```
                }

        }
}

// <=
bool Orless(int* number1, const int sizel1, int* number2, const int
sizel2)
{
        if (Oequal(number1, sizel1, number2, sizel2))
                return true;
        else
        {
                if (sizel1 != sizel2)
                        return sizel1 < sizel2;
                else
                {
                        for (int i = 0; i < sizel1; i++)
                        {
                                if (number1[i] !=
number2[i])
                                        return
number1[i] < number2[i];
                        }
                }

        }
}

// >=
bool Obig(int* number1, const int sizel1, int* number2, const int
sizel2)
{
        if (!Oless(number1, sizel1, number2, sizel2)) // Значит что
>=
        {

                return true;

        }

        return false;
}

/* Арифметика */

// Сумма чисел
void Addition(int* number1, const int sizel1, int* number2, const int
sizel2, ListH& result, int& sizelr)
{
        OPEN_LOG
        log << "Производим сложение: \n";
        sizelr = max(sizel1, sizel2) + 1;
        int* numberR = new int[sizelr];
        // Меняем элементы местами
        ReverseInt(number1, sizel1);
        ReverseInt(number2, sizel2);

        log << "Числа переворачиваем для удобства: " <<
Out(number1, sizel1) << endl << Out(number2, sizel2) << endl;


        int carry = 0, total = 0;
        for (int i = 0; i < sizelr; i++)
        {
                // Достаем цифры, чтобы не дойти случайно
до конца массива.
                int digit1 = i < sizel1 ? number1[i] : 0;
                int digit2 = i < sizel2 ? number2[i] : 0;

                // Считаем
                total = digit1 + digit2;

                // Если число+ остаток больше 9 то в ответ
пришем посл цифру если нет то просто число+остаток
                numberR[i] = total + carry > 9 ? (total + carry) %
10 : total + carry;
```

```
        ListH(const ListH& other);


        void push_back(String& str);
        void copyList(const ListH& lst);


        int GetSize();

        String& operator[](int index);

        friend std::ostream& operator<<(std::ostream& os, ListH&
listH);



        void pop(int n);
        ~ListH();

};
```

ListV.cpp

```
#include "ListV.h"

ListV::ListV()
{
        head = cur = last = nullptr;
        size = 0;
}

void ListV::push_back(ListH& list)
{
        if (head == nullptr)
        {
                head = new NodeV(list);
                last = head;
                size++;
        }
        else
        {
                NodeV* tmp = new NodeV(list, last);
                last->lstNext = tmp;
                last = tmp;
                size++;
        }
}

ListH& ListV::operator[](int index)
{
        if (index > size)
        {
                throw std::out_of_range("Index Out Of Range");
        }

        cur = head;
        for (int i = 0; i < index; i++)
        {
                cur = cur->lstNext;
        }

        return cur->elemH;
}

void ListV::pop_back()
{
        if (size <= 0)
                return;

        NodeV* tmp = last->lstPrev;

        delete last; size--;

        if (size == 0)
                return;

        tmp->lstNext = nullptr;
```

```
			log << "После " << i << "-го сложения получим: " << Out(numberR, i) << endl;

			// Сохраняем остаток
			carry = (total + carry) / 10;

		}

		// Удаляем ненужный ноль если он есть.
		if (numberR[sizelr - 1] == 0)
			sizelr--;


		// Переворачиваем числа обратно
		ReverseInt(number1, sizel1);
		ReverseInt(number2, sizel2);
		ReverseInt(numberR, sizelr);


		MakeList(result, numberR, sizelr);

		log << "В результате: " << Out(numberR, sizelr) << endl;

		delete[] numberR;
}

// Вычитание
void Sub(int* number1, const int sizel1, int* number2, const int sizel2, ListH& result, int& sizelr)
{
		OPEN_LOG
		log << "Производим вычитание: \n";
		sizelr = max(sizel1, sizel2);
		int* numberR = new int[sizelr];

		// Меняем элементы местами
		ReverseInt(number1, sizel1);
		ReverseInt(number2, sizel2);
		log << "Числа переворачиваем для удобства: " << Out(number1, sizel1) << endl << Out(number2, sizel2) << endl;


		// Создаем коппию первого числа на всякий случай
		int* number1copy = new int[sizel1];
		for (int i = 0; i < sizel1; i++)
		{
			number1copy[i] = number1[i];
		}

		// Определяем знак ответа.

		int carry = 0;
		for (int i = 0; i < sizelr; i++)
		{
			// Достаем цифры, чтобы не дойти случайно до конца массива.
			int digit1 = i < sizel1 ? number1copy[i] : 0;
			int digit2 = i < sizel2 ? number2[i] : 0;

			// Считаем
			int total = digit1 - digit2;

			if (total < 0)
			{
				number1copy[i + 1] -= 1;
				total += 10;
			}


			numberR[i] = total;

			log << "После " << i << "-го вычитания получим: " << Out(numberR, i) << endl;
		}

		int i = sizelr;
		while (numberR[i - 1] == 0)
		{
			last = tmp;
}

// Удаление элемента по индексу.
void ListV::pop(int n)
{
		if (n >= size-1 || (n==0 && size==1))
		{
			pop_back();
		}
		else
		{
			if (size == 0)
				return;

			cur = head;
			for (int i = 0; i < n; i++)
			{
				cur = cur->lstNext;
			}


			cur->lstNext->lstPrev = cur->lstPrev;
			if (cur != head)
				cur->lstPrev->lstNext = cur->lstNext;
			else
				head = cur->lstNext;

			delete cur; size--;


		}
}

void ListV::OutList(std::ostream& os)
{
		int n = 0;
		cur = head;
		while (cur != nullptr)
		{
			os << BLUE <<++n << RESET << ". " << cur->elemH << std::endl;
			cur = cur->lstNext;
		}
}


void ListV::push(ListH& list, int pos)
{
		if (pos > 0 && pos < size)
		{
			// Доходим до элемента,  потом просто запихиваем строку.

			cur = head;
			for (int i = 0; i < pos; i++)
			{
				cur = cur->lstNext;
			}

			cur->elemH.copyList(list);
		}
		else
		{
			push_back(list);
		}
}

ListV::~ListV()
{
		if (size != 0)
		{
			cur = head;
			while (cur != nullptr)
			{
				NodeV* tmp = cur->lstNext;
				delete cur;
```

```
			sizelr--;
			i--;
		}

		/*OutNumber(number1, sizel1);*/

		// Переворачиваем числа обратно
		ReverseInt(number1, sizel1);
		ReverseInt(number2, sizel2);
		ReverseInt(numberR, sizelr);


		log << "В результате: " << Out(numberR, sizelr) << endl;

		MakeList(result, numberR, sizelr);

		delete[] number1copy;
		delete[] numberR;
		CLOSE_LOG
}


// Умножение
void Multiply(int* number1, const int sizel1, int* number2, const int
sizel2, ListH& result, int& sizelr)
{
		OPEN_LOG
		// Если умножаем на 0.
		if (sizel1 == 1 && number1[0] == 0 || sizel2 == 1 &&
number2[0] == 0)
		{
			int* zero = new int[1];
			zero[0] = 0;
			MakeList(result, zero, 1);
			return;
		}


		// Выделение памяти для результата умножения.
		sizelr = sizel1 + sizel2;
		int* numberR = new int[sizelr]();
		// Для промежуточных результатов.
		int promSize = sizel1 + 1;
		const int begsize = promSize;
		int* promNumber = new int[promSize];
		for (int i = 0; i < sizel1 + 1; i++)
		{
			promNumber[i] = numberR[i];
		}
		promNumber[promSize - 1] = 0;

		int w = sizelr; // Сдвиг
		// Выводим промежуточные результаты.
		for (int i = sizel2 - 1; i >= 0; i--) // number 2
		{
			// Стандартное умножение
			int carry = 0;
			int all = promSize - 1;
			for (int j = sizel1 - 1; j >= 0; j--) // number 1
			{
				int cur = number2[i] * number1[j] +
promNumber[all] + carry;

				promNumber[all--] = cur % 10;

				carry = cur / 10;

			}

			promNumber[all] += carry;


			// Вывод числа.
			RemoveZeroes(promNumber, promSize);

			if (promNumber[0] != 0)
			{
				ListH tempLst;
```

```
				cur = tmp;
			}
		}
		head = cur = last = nullptr;
		size = 0;
}
```

ListV.h

```
#pragma once
#include "NodeV.h"

class ListV
{
public:
		NodeV* head;
		NodeV* cur;
		NodeV* last;
		int size;

		ListV();
		~ListV();

		void push_back(ListH& list);

		ListH& operator[](int index);
		void pop_back();
		void pop(int n);

		void OutList(std::ostream& os);


		void push(ListH& list, int pos); // Сохранить элемент в
определенное место.

};
```

NodeH.cpp

```
#include "NodeH.h"

NodeH::NodeH(String& str, NodeH* strPrev, NodeH* strNext)
{
		this->elem = str;
		this->strPrev = strPrev;
		this->strNext = strNext;
}

NodeH::~NodeH()
{
}
```

NodeH.h

```
#pragma once
#include "String.h"

class NodeH
{
public:
		String elem;
		NodeH* strPrev;
		NodeH* strNext;


		NodeH(String& str, NodeH* strPrev=nullptr, NodeH*
strNext=nullptr);
		~NodeH();
};
```

NodeV.cpp

```
#include "NodeV.h"

NodeV::NodeV(ListH& elemH, NodeV* strPrev, NodeV* strNext)
{
		this->elemH.copyList(elemH);
		this->lstPrev = strPrev;
		this->lstNext = strNext;
}
```

```
                        MakeList(tempLst, promNumber,
promSize);

                        cout << '\t' << setw(w) << tempLst <<
endl;
                        log << '\t' << setw(w) << tempLst <<
endl;

                        tempLst.~ListH();
                }
                w--;

                // Очистка числа.
                for (int i = 0; i < begsize; i++)
                {
                        promNumber[i] = 0;
                }
                promSize = begsize;

        }



        for (int i = sizel2 - 1; i >= 0; i--) // number 2
        {
                int carry = 0;
                for (int j = sizel1 - 1; j >= 0; j--) // number 1
                {
                        int cur = number2[i] * number1[j] +
numberR[i + j + 1] + carry;

                        numberR[i + j + 1] = cur % 10;

                        carry = cur / 10;

                }

                if (carry != 0)
{
                        numberR[i] += carry;
                }
        }

        // Удаление ведущих нулей.
        RemoveZeroes(numberR, sizelr);



        MakeList(result, numberR, sizelr);

        delete[] numberR;

        CLOSE_LOG
}
```

String.cpp

```
#include "String.h"

String::String(int len): len(len)
{

}

String::~String()
{
}

int String::Len()
{
        return len;
}

char& String::operator[](int index)
{
        if (index > len)
                throw std::out_of_range("Index Out Of Range");
```

```
NodeV::~NodeV()
{
}
```

NodeV.h

```
#pragma once
#include "ListH.h"

class NodeV
{
public:
        ListH elemH;
        NodeV* lstPrev;
        NodeV* lstNext;

        NodeV(ListH& elemH, NodeV* lstPrev = nullptr, NodeV*
lstNext = nullptr);
        ~NodeV();
};
```

String.h

```
#pragma once

#define N 5
#define RED "\033[31m"
#define GREEN "\033[32m"
#define BLUE "\033[34m"
#define YELLOW "\033[33m"
#define CYAN "\033[36m"
#define RESET "\033[0m"


#include <stdexcept>
#include <ostream>


class String
{
        char str[N];
        int len = N;
public:
        //Constructors
        String(int len=N);
        ~String();

        //Getters
        int Len();
        char& operator[](int index);

        //Setters
        void SetElem(char elem, int index);
        void SetLen(int newlen);

        int Znach();


        //Methods
        void Clear();

        friend std::ostream& operator<<(std::ostream& os, String&
string);

};
```

```
	else
		return str[index];
}

void String::SetElem(char elem, int index)
{
	if (index < len)
		str[index] = elem;
}

void String::SetLen(int newlen)
{
	if (newlen > N)
		len = N;
	else
		len = newlen;
}

int String::Znach()
{
	return atoi(str);
}

void String::Clear()
{
	for (int i = 0; i < len; i++)
	{
		str[i] = '\0';
	}
	len = N;
}

std::ostream& operator<<(std::ostream& os, String& string)
{
	char* newChar = new char[string.len + 1];
	for (int i = 0; i < string.len; i++)
	{
		newChar[i] = string[i];
	}
	newChar[string.len] = '\0';

	os << newChar;

	delete[] newChar;

	return os;
}
```

Пример работы программы.

Пример считывания №1

```
123
12
123
699110
52
123456789
987654321
1234
13
728145
97652851
100000371
200252352525
15612352525
865802619705879147149759275
0829572097509285920957-95709752
```

```
+ 1 2
/ 8 9 10
/ 4 5
/ 1 3
* 10 11 11
/ 10 3 9
* 1 2
- 15 16
/ 5 4
```

```
Файл успешно считан.

Полученный список:
1. 123
2. 12
3. 123
4. 699110
5. 52
6. 123456789
7. 987654321
8. 1234
9. 13
10. 728145
11. 97652851
12. 100000371
13. 200252352525
14. 15612352525
15. 865802619705879147149759275
16. 0829572097509285920957-95709752

Введи нужный номер запроса:
1. Число под номером 1 + Число под номером 2
2. Число под номером 8 / Число под номером 9
3. Число под номером 4 / Число под номером 5
4. Число под номером 1 / Число под номером 3
5. Число под номером 10 * Число под номером 11
6. Число под номером 10 / Число под номером 3
7. Число под номером 1 * Число под номером 2
8. Число под номером 15 - Число под номером 16
```

Пример считывания №2

```
562856825981581
915491891
5818501
12
0
5810481
100000000
6781
```

```
+ 1 2
* 3 4
- 6 4 1
+ 2 3
/ 1 4
```

```
Файл успешно считан.

Полученный список:
1. 562856825981581
2. 915491891
3. 5818501
4. 12
5. 0
6. 5810481
7. 100000000
8. 6781

Введи нужный номер запроса:
1. Число под номером 1 + Число под номером 2
2. Число под номером 3 * Число под номером 4
3. Число под номером 6 - Число под номером 4
4. Число под номером 2 + Число под номером 3
5. Число под номером 1 / Число под номером 4
0. Назад.
```

Примеры операций:

Деление:

```
Введите номер: 4

Результат деления чисел:

       123|123
       Ответ: 1
Результат сохранен в конец списка!
```

```
Введите номер: 5

Результат деления чисел:

       728145|123
       615   |
       -------
       1131
       1107
         244
         123
         1215
         1107
       Ответ: 5919
Результат сохранен на 9 место списка.
```

Умножение:

```
Результат умножения чисел:

                562856825981581
                     915491891
       ------------------------
                562856825981581
              5065711433834229
             4502854607852648
             562856825981581
            5065711433834229
           2251427303926324
          2814284129907905
          562856825981581
        5065711433834229
        ------------------------
        515290859980135520859671

Результат сохранен в конец списка!
```

Сложение:

```
Результат сложения чисел:

        562856825981581
              915491891
        ---------------
        562857741473472

Результат сохранен в конец списка!
```

Вычитание:

```
Результат вычитания чисел:

        5810481
             12
        -------
        5810469

Результат сохранен в конец списка!
```

Выводы.

В ходе курсовой работы я закрепил все знания языка C++ полученные мною во втором семестре, научился организовывать линейные списки, реализовал длинную арифметику с использованием этих списков.