Отчет по лабораторной работе № 1 по дисциплине

“Типы и структуры данных”

Работу выполнил:

Вариант 3

2019 г.

Отчет

Постановка задачи:

Реализация арифметических операций над числами, выходящими за разрядную сетку персонального компьютера, выбор необходимых типов данных для хранения и обработки указанных чисел.

Описание условия задачи:

Смоделировать операцию умножения целого числа длиной до 30 десятичных цифр на действительное число в форме ±m.n Е ±K, где суммарная длина мантиссы (m+n) - до 30 значащих цифр, а величина порядка K - до 5 цифр. Результат выдать в форме ±0.m1 Е ±K1, где m1 - до 30 значащих цифр, а K1 - до 5 цифр.

Описание ТЗ:

1.Введение:

1.1 Наименование программы:

Умножение целого и действительного числа размером до 30 знаков.

### 1.2. Краткая характеристика области применения:

Умножение чисел большой̆ размерности при выполнении астрономических расчетов.

### 1.3. Срок выполнения работы:

2 недели с момента выдачи задачи.

2. Основания для разработки:

### 2.1. Заказчик

Силантьева Александра Васильевна

### 2.2. Исполнитель

Исполнитель – Савинов Егор Дмитриевич, студент группы ИУ7-34б.

### 2.3. Основания для разработки

Учебная программа 3 семестра программы ИУ7.

## 3. Назначение разработки

3.1 Общая концепция системы:

Составить программу умножения целого и вещественного числа, где у вещественного порядок имеет до 5 знаков: от –99999 до +99999, а мантисса – до 30 знаков, а длина целого числа до 30 знаков. Программа должна осуществлять ввод чисел и выдавать либо верный результат в указанном формате (при корректных данных), либо сообщение о невозможности произвести счет.

* 1. Описание функциональности системы:

Умножает вещественное и целое число больших размеров.

## 4. Требования к программе:

4.1 Требования к информационным структурам:

Программа должна уметь отрабатывать любой ввод пользователя. При корректных данных следует выводить результат в нормализованном виде, иначе нужно выдавать сообщение об ошибке и после этого программа должна завершить свою работу.

### 4.2. Требования к функциональным характеристикам:

Программа должна выполнять:

1. Умножать вещественное и целое число.
2. Записывать ответ в нормализованном виде.
3. Если мантисса переполнена, то округлять до 30 значащих значений.
4. Обрабатывать ввод чисел.
5. Обрабатывать переполнение.

Сначала программа предлагает ввести целое число вида: ±m, затем вещественное число вида ±m.n E ±K (суммарная длина мантиссы (m+n) - до 30 значащих цифр, а величина порядка K - до 5 цифр). Результат должен быть записан в нормализованном виде: ±0.m1 E±K1. Программа должна работать при корректных данных ввода, при возникновении ошибки должна вывести на экран тип ошибки и завершить свою работу.

### 4.3. К надежности:

1. Программа должна уметь обрабатывать любой ввод, также при введении некорректных данных не должна завершаться аварийно.

2. Программа должна считывать числа вида: ±m, ±m E±K, ±m.n, ±m.n E±K.

3. Если вычисления невозможны, программа должна завершиться не аварийно.

4.4 Требования к составу аппаратного обеспечения:

У клиента интегрированная среда разработки должна поддерживать файлы с расширением .с. Программа не требует поддержки со стороны пользователя.

## 5. Стадии и этапы разработки:

Стадии:

1. Начальный этап разработки (ввод чисел и защита от некорректного ввода)
2. Тестирование умножения числа на число.
3. Разработка вывода программы.
4. Тестирование программы.
5. Доработка на основе тестирования программы и сдача окончательного варианта программы лабораторной работы.

Этапы:

Получение от преподавателя лабораторной работы и сроков сдачи программы, написание тестов для программы, создание программы, тестирование, написание отчета по проделанной работе, сдача программы и отчета.

6. Описание структуры данных:

1.Переменные типа int:  
int smejenie = 0; // счетчик смещения при умножении «столбиком»  
int tek\_cislo = 0; // результат умножения числа на число   
int k = 0; // счетчик смещения при записи результата  
int flag\_poradok = 0; // флаг на отрицательный порядок  
int poradok = 0; // порядок числа  
int n\_float\_poradok = 0; // переменная для корректировки порядка в соответствии с результатом (считает длину порядка при нормализации начального вещественного числа)

int flag\_e; // флаг на появление в числе «Е»

int flag\_pointer; // флаг на появление в числе «.»

int n\_int // длина целого числа.

int n\_float // длина вещественного числа.

2. Массивы типа \*char:

char poradok\_char[6]; // хранение порядка в виде набора символов

char \*int\_num // хранение целого числа в виде набора символов

char\* float\_num // хранение вещественного числа в виде набора символов

3. Массивы типа \*int:

int mantissa[MAX\_MANTISSA\_N] = {0}; // используется для хранения результата вычислений.

7. Описание алгоритма:

1. int input\_int(char \*int\_num):

Функция сначала выводит инструкцию ввода целого числа. Далее просит пользователя ввести целое число с указанием знака числа. После этого функция проверяет число на длину, разрешенные символы и подсчитывает и возвращает длину числа. Если число введено неверно, выводит соответствующий код ошибки и завершает программу. Функция посимвольно записывает каждый символ в соответствующий массив для дальнейшей обработке в функции int multiplication\_of\_number().

2. int input\_float (char \*float\_num):

Функция сначала выводит инструкцию ввода вещественного числа. Далее просит пользователя ввести вещественное число вида: ±m, ±m E±K, ±m.n, ±m.n E±K. После этого функция проверяет число на длину мантиссы, порядка, разрешенные символы и подсчитывает и возвращает длину числа. Если число введено неверно, выводит соответствующий код ошибки и завершает программу. Функция посимвольно записывает каждый символ в соответствующий массив для дальнейшей обработке в функции int multiplication\_of\_number().

3. int multiplication\_of\_number(int n\_int, int n\_float, char \*int\_num, char\* float\_num, int flag\_e, int flag\_pointer):

Функция сначала считает порядок вещественного числа. Далее происходит умножение столбиком: во вложенном цикле каждый символ целого и вещественного числа переводится в тип int и перемножается. Далее если результат больше 9, то в массиве результата на соответствующую позицию (с учетом смещения и количества умножений) прибавляем остаток от деления на 10 и число, которое находилось в “уме”(целую часть от переполнения предыдущих результатов), если в свою очередь и этот результат больше 9, то в ячейку результата помещается остаток от деления результата на 10, а целая часть от деления уходит в следующую переменную (“i – 1”). Если при перемножении символа вещественного и целого числа, результат меньше 9, то используется все тот же алгоритм, но уже прибавляем к соответствующей позиции не остаток от деления на 10, а само число.

Далее корректируется порядок числа: складывается порядок введенного вещественного числа и длина получившегося результата, если длина вещественного числа меньше длины получившегося результата.

Далее определяется по знаку мантиссы вещественного числа и целого числа, знак результата мантиссы нормализированной формы, было ли одно из введенных чисел нуль, если да, то записывается результат “0”.

Также на этом этапе происходит проверка длины мантиссы числа(если больше 30, то идет округление по правилам математики). Если ошибок не обнаружено, то удаляются незначащие нули и записывается результат работы умножения чисел.

8. Тесты:

|  |  |  |
| --- | --- | --- |
| название | ввод | вывод |
| Пустая строка | “ ” | ERROR: Длина числа должна быть больше нуля |
| Наличие некорректных символов при вводе | “+12a” | ERROR: Некорректный ввод цифр целого числа |
| Ввод маленькой «e» | “+12” “12.e+10” | ERROR: Некорректный ввод вещественного числа |
| Превышение длины целого числа | «+1111111111111111111111111111111» | ERROR: Длина числа превышает 30 десятичных цифр |
| Превышение мантиссы | «+12» «+1111111111111111111111111111111Е+10» | ERROR: Некорректный ввод вещественного числа |
| Ввод двух точек | «+12» «12.1.1» | ERROR: Некорректный ввод вещественного числа |
| Превышение порядка | «+12» «12E+100000» | ERROR: Длина порядка вещественного числа превышает 5 символов |
| Незначащие нули | «+000100» «+0.01» | +0.1E+1 |
| Умножение на ноль | «+0» «+100E+100» | 0 |
| Умножение на единицу | «+1» «+0.0001» | +0.1E-3 |
| Умножение чисел разных знаков | «-100» «+0.01» | -0.1E+1 |
| Переполнение порядка | «+99» «+999.999E+99999» | переполнение порядка |
| Округление мантиссы | «+2» «+0.999999999999999999999999999999» | +0.2E+1 |
| Умножение отрицательных чисел | «-100» «-1Е-5» | +0.1E-2 |
| Проверка верхней границы порядка (1) | «+10» «+1E+99999» | переполнение порядка |
| Проверка верхней границы порядка (2) | «+10» «+0.01E+99999» | +0.1E+99999 |
| Проверка верхней границы порядка (3) | «+1» «+0.01E+99999» | +0.1E+99998 |
| Проверка умножения на максимальные символы | «+999» «+999» | +0.998001E+6 |
| Проверка нижней границы порядка (1) | «+10» «+1E-99999» | +0.1E-99997 |
| Тест на «9» | «+999999999999999999999999999999»  «+100000000000000000000000000001» | +0.100000000000000000000000000001E+60 |
| Проверка нижней границы порядка (2) | «+1» «+0.01E-99999» | переполнение порядка |

Описание аварийных ситуаций:

Некорректный ввод:

* Ввод точек, букв, нескольких знаков «+» и «-» пробелы при вводе целочисленного числа.
* Ввод букв, нескольких знаков «+» и «-», «.», «Е», маленькой «е», пробелы при вводе вещественного числа.

Программа выведет, что было введено некорректное число и завершит свою работу.

Переполнение:

При вычислении порядка может возникнуть его переполнение (больше +99999 или меньше -99999).

Программа выведет, что при вычислении порядок переполнился и завершит свою работу.

Таким образом, программа если не может выдать корректный ответ, выводит сообщение об характере ошибки и завершает свою работу.

9. Выводы:

Данная лабораторная работа показалась сложна для реализации. Алгоритм умножения целого и вещественного числа требует сложного унифицирования для работы со всеми возможными типами вещественных чисел.

Наверное, при реализации динамического выделении памяти, программа работала бы немного быстрее.

10. Вопросы:

10.1 Каков возможный диапазон чисел, представляемых в ПК?

Если под хранение целого положительного числа выделено 16 разрядов, то его максимальное значение не может превышать 2^16 - 1=65535, если выделено 32 разряда, то максимальное значение составит 2^32 - 1=4 294 967 295. Для 64 разрядов максимально возможное значение числа равно 2^64 - 1=18 446 744 073 709 551 615.

10.2 Какова возможная точность представления чисел, чем она определяется?

Вещественно число хранится в виде (-1)^S \* M \* B^E, где S — знак, B — основание, E — порядок, а M — мантисса ( правильная дробь в интервале [0.1..1)).Точность представления числа зависит от длины мантиссы, а длина мантиссы зависит от наличия знака и длины выделяемой памяти. Если мантисса выходит за рамки предоставленной памяти, то число будет округлено. На точность влияет битность системы: если 64 бит, то на порядок 11 бит и на знак 1 бит, тогда на мантиссу остается 64 - 11 – 1 = 52 бита, следовательно для 52 разрядов максимально возможное значение числа равно 2^52 - 1= 4 503 599 627 370 496.

10.3 Какие стандартные операции возможны над числами?

Умножение, деление, сложение, вычитание, сравнение.

10.4 Какой тип данных может выбрать программист, если обрабатываемые числа превышают возможный диапазон представления чисел в ПК?

Может выбрать массив.

10.5 Как можно осуществить операции над числами, выходящими за рамки машинного представления?

Можно записать цифры числа в массив и дальше производить операцию поэлементно. Порядок также считать отдельно.