Пояснювальна записка

Зміст

[Вступ 3](#_Toc342605741)

[1 Теоретичні відомості 4](#_Toc342605742)

[1.1 Форми подання чисел в ЕОМ 4](#_Toc342605743)

[1.2 Операції з числами у форматі з плаваючою комою 5](#_Toc342605744)

[1.2.1 Додавання чисел з плаваючою комою 5](#_Toc342605745)

[1.2.2 Множення чисел з плаваючою комою 5](#_Toc342605746)

[1.2.3 Ділення чисел із плаваючою комою 5](#_Toc342605747)

[2 Проектування бібліотеки 6](#_Toc342605748)

[2.1 Функції бібліотеки для чисел з плаваючою комою 6](#_Toc342605749)

[2.2 Документація бібліотеки 7](#_Toc342605750)

[Висновки 17](#_Toc342605760)

[Список інформаційних джерел 18](#_Toc342605761)

[Додаток А. Програмний код роботи 19](#_Toc342605762)

Вступ

У курсовій роботі розробляється програмна бібліотека функцій, що дозволяє виконувати обчислення над числами з плаваючою комою, а саме: арифметичні операції, увід із рядка і перетворення в строкову форму.

Даний програмний модуль призначений для підтримки виконання операцій з плаваючою комою в процесорах, які можуть виконувати операції тільки з цілими числами. Прикладами таких процесорів є “integer processors” ST220 і ST231 фірми STMicroelectronics.

# Теоретичні відомості

## Форми подання чисел в ЕОМ

Будь-яке число X в позиційній системі числення з основою k можна записати як



де *M* – мантиса числа *X*, *P* – порядок числа X.

Якщо кожне число задається парою P і M, то таке подання називають формою із плаваючою комою.

За подання чисел у формі із плаваючою комою порядок P може бути додатнім або від’ємним цілим числом. Мантиса ж у більшості випадків є додатнім або від’ємним правильним дробом, причому



Це означає, що старший розряд модуля мантиси завжди дорівнює 1 (мантиса нормалізована). Якщо в ЕОМ для запису порядку використовується m розрядів, а для запису мантиси – n розрядів, то в цій машині може бути подане таке максимальне (за абсолютною величиною) число:



У свою чергу, мінімальне за абсолютною величиною число, що відрізняється від нуля, у такій машині дорівнює



Крім зазначених вище розрядів, для подання порядку й мантиси у розрядній сітці ЕОМ із плаваючою комою є також розряди для подання порядку й мантиси.

Для універсальних комп’ютерів розроблений стандарт ANSI/IEE 754-985 на подання чисел з плаваючою комою.

## Операції з числами у форматі з плаваючою комою

### Додавання чисел з плаваючою комою

Суму двох чисел  і , поданих у форматі із плаваючою комою, можна записати у вигляді



де за  прийнято перетворену мантису числа з меншим порядком,.

Етапи додавання чисел з плаваючою комою:

1. зрівняння порядків;
2. додавання мантис;
3. нормалізація результату і формування результату.

### Множення чисел з плаваючою комою

Множення двох чисел  і , поданих у форматі із плаваючою комою, можна записати у вигляді



Етапи множення чисел з плаваючою комою:

1. одержання порядку результату у вигляді суми порядків множників;
2. знаходження мантиси результату;
3. нормалізація результату.

### Ділення чисел із плаваючою комою

Результат ділення двох чисел  і , поданих у форматі із плаваючою комою, можна записати у вигляді



Ділення мантис має виконуватися за умови .

Етапи ділення чисел із плаваючою комою:

1. одержання порядку результату шляхом віднімання;
2. одержання мантиси результату. шляхом ділення мантис;
3. нормалізація результату.

# Проектування бібліотеки

Тип даних з плаваючою комою реалізується у вигляді структури за допомогою оператора перетворення типів typedef. Структура містить два поля: char mantiss для представлення мантиси і char exp – для порядку.

Представлення числа з плаваючою комою за допомогою рядка здійснюється в будь-якій системі числення послідовним виділенням розрядів числа (алгоритм аналогічний переводу чисел в іншу систему числення).

Увід числа за допомогою рядка виконується за допомогою послідовного множення значення розряду на основу розряду і додаванням до суми розрядів, які вже є.

## Функції бібліотеки для чисел з плаваючою комою

reduction Приведення до одного порядку двох чисел з плаваючою комою.

div Ділення двох чисел з плаваючою комою.

mult Множення двох чисел з плаваючою комою.

sub Віднімання двох чисел з плаваючою комою.

add Додавання двох чисел з плаваючою комою.

quotient Цілочисельне ділення.

rest Залишок від ділення.

init Ініціалізація початковим значенням.

dblcmp Порівняння двох чисел з плаваючою комою.

asc2dbl Перетворює рядок символів в число з плаваючою комою.

dblasc2 Перетворює число з плаваючою комою в рядок символів.

## Документація бібліотеки

FLIP

**File List**

Here is a list of all files with brief descriptions:

|  |  |
| --- | --- |
| o*[**adduction.cpp**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\adduction_8cpp.html) | Код для [**adduction.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\adduction_8h.html) |
| o[*](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\adduction_8h_source.html)[**adduction.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\adduction_8h.html) | Визначена функція для вирівнювання порядків двох чисел типу My\_Double |
| o*[**flipunittest.cpp**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\flipunittest_8cpp.html) | Тестування бібліотеки "FLIP" за допомогою **MSTest** |
| o*[**mdb\_manipule.cpp**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8cpp.html) | Код для [**mdb\_manipule.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8h.html) |
| o[*](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8h_source.html)[**mdb\_manipule.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8h.html) | Основні маніпуляції з числом типу My\_Double |
| o*[**mdb\_string.cpp**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8cpp.html) | Код для [**mdb\_string.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h.html) |
| o[*](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h_source.html)[**mdb\_string.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h.html) | Основні маніпуляції з рядком і перетворення рядка в число типу My\_Double |
| o[*](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h_source.html)[**my\_double.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html) | Представлення числа з плаваючою точкою в пам'яті |
| o*[**operations.cpp**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8cpp.html) | Код для [**operations.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html) |
| \[*](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h_source.html)[**operations.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html) | Арифметичні операції над числами з плаваючою крапкою |

**Data Structures**

Here are the data structures with brief descriptions:

|  |  |
| --- | --- |
| oN[**UnitTestFLIP**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html) |  |
| \C[**my\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\structmy___double.html) | Структура, що представляє число з плаваючою точкою |

**UnitTestFLIP Namespace Reference**

|  |  |  |
| --- | --- | --- |
| Functions | | |
|  | [**TEST\_CLASS**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#a7dec957cc4710c58327c6a50f64521f0) (UnitTestStringToMyDouble) | |
|  | Тестує функцію для переведення рядка символів в число типу My\_Double. | |
|  | | |
|  | [**TEST\_CLASS**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#ae6f45e817b2453923ed434c8bbaeb714) (UnitTestAdding) | |
|  | Тестує операцію додавання | |
|  | | |
|  | [**TEST\_CLASS**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#a5ef73b190d6f190506a9bd61f7f2cff3) (TestDivision) | |
|  | Тестує операцію ділення. | |
|  | | |
|  | [**TEST\_CLASS**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#a6271d17629a314f22ed395e3a706205f) (TestMultiply) | |
|  | Тестує операцію множення. | |
| 1. Variables | | |
| int | | [**OVER\_FLOW**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#a2e20158957f9d02a6aeda6b467f9b826) = -1 |
|  | | |
| int | | [**WRONG\_SYMBOL**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#ab56c7f99f307a6dc6fd5b1350f2b0840) = -2 |
|  | | |
| int | | [**WRONG\_PARAMETER**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\namespace_unit_test_f_l_i_p.html#a8e15167acdf2d23eb0fd7e5109bcfef1) = -3 |
|  | | |

Function Documentation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UnitTestFLIP::TEST\_CLASS** | **(** | **UnitTestStringToMyDouble** |  | **)** |  |

Тестує функцію для переведення рядка символів в число типу My\_Double.

Перевіряється запис рядка "1.0" в число типу My\_Double.

Переповнення типу при вводі надто великого рядка.

Спроба записати рядок із недопустимими для даної системи числення символами.

Спроба записати число з двоми десятковими точками.

Мінус в неправильній позиції.

У числа два мінуса.

Неправильне поєднання символів.

Невірні символи в рядку.

Основа системи числення менша, ніж два.

Основа системи числення більша, ніж шістнадцять.

Нульовий вказівник в якості числа.

Нульовий вказівник в якості рядка.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UnitTestFLIP::TEST\_CLASS** | **(** | **UnitTestAdding** |  | **)** |  |

Тестує операцію додавання

Перевірка операції 1+(-1).

Перевірка суми 1+(-0.001).

Переповнення мантиси зверху INT\_MAX\*2^10 + INT\_MAX\*2^10.

Переповнення мантиси знизу -INT\_MAX\*2^2 - INT\_MAX\*2^2.

INT\_MAX\*2 - INT\_MAX/2.

Переповнення порядку зверху INT\_MAX\*2^INT\_MAX + 1.

Переповнення порядку знизу -INT\_MAX\*2^INT\_MAX-1.

Запис в нульовий вказівник.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UnitTestFLIP::TEST\_CLASS** | **(** | **TestDivision** |  | **)** |  |

Тестує операцію ділення.

Перевірка операції 1/2 = 0.5 = 1\*2^(-1).

Переповнення мантиси INT\_MAX\*2 / (INT\_MAX/2) = 4.

Переповнення порядку зверху (INT\_MAX\*2^INT\_MAX)/(1/2).

Переповнення порядку знизу 2^(-INT\_MAX)/2.

Ділення на нуль.

Запис в нульовий вказівник.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UnitTestFLIP::TEST\_CLASS** | **(** | **TestMultiply** |  | **)** |  |

Тестує операцію множення.

Перевірка операції -2\*24 = -48.

Переповнення порядку зверху 2^INT\_MAX \* 2.

Переповнення порядку знизу 2^(-INT\_MAX) \* 1/2.

Множення на нуль.

Запис в нульовий вказівник.

**adduction.h File Reference**

Визначена функція для вирівнювання порядків двох чисел типу My\_Double.

#include "[**my\_double.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h_source.html)"

|  |  |
| --- | --- |
| 1. Functions | |
| void | [**reduction**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\adduction_8h.html#aaf5f511f008f578f6b3f45a2eda7374a) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*example, [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*pattern) |
|  | Функція вирівнює порядки двох чисел. Можлива втрата точності. |
|  | |

Detailed Description

Визначена функція для вирівнювання порядків двох чисел типу My\_Double.

Function Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **void reduction** | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **example,** |
|  |  | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **pattern** |
|  | **)** |  |  |

Функція вирівнює порядки двох чисел. Можлива втрата точності.

**Parameters**

|  |  |
| --- | --- |
| **example** | вказівник на число типу My\_Double, до порядку якого приводять. Не може бути пустим вказівником. |
| **pattern** | вказівник на число типу My\_Double, порядок якого треба зрівняти з example. Не може бути пустим вказівником. |

**mdb\_manipule.h File Reference**

Основні маніпуляції з числом типу My\_Double.

#include "[**my\_double.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h_source.html)"

|  |  |
| --- | --- |
| 1. Functions | |
| void | [**init**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8h.html#a77cc6205d4af95ab351154c7f863213b) (int num, [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*number) |
|  | Функція ініалізує змінну number цілим числом num. Можлива втрата точності. |
|  | |
| int | [**dblcmp**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8h.html#aacb649b07d937ab1419c6d389fc9de48) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) left, [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) right) |
|  | Функція порівнює дві змінних типу My\_Double. |
|  | |
| void | [**mdbcpy**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__manipule_8h.html#a0b00b85c2334e51a84e83a12ff7145bf) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*duplicate, const [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*pattern) |
|  | Функція копіює вміст вказівника pattern в duplicate. |
|  | |

Detailed Description

Основні маніпуляції з числом типу My\_Double.

Function Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **int dblcmp** | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) | **left,** |
|  |  | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) | **right** |
|  | **)** |  |  |

Функція порівнює дві змінних типу My\_Double.

**Parameters**

|  |  |
| --- | --- |
| **left** | змінна типу My\_Double. |
| **right** | змінна типу My\_Double. |

**Returns**

0, якщо числа рівні.

1, якщо left більше right.

-1, якщо left менше right.

|  |  |  |  |
| --- | --- | --- | --- |
| **void init** | **(** | **int** | **num,** |
|  |  | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **number** |
|  | **)** |  |  |

Функція ініалізує змінну number цілим числом num. Можлива втрата точності.

**Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **num** | | ціле число. | | | |
| **number** | | змінна типу My\_Double. Не може бути пустим вказівником. | | | |
| **void mdbcpy** | | **(** | | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **duplicate,** |
|  | |  | | **const**[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **pattern** |
|  | | **)** | |  |  |

Функція копіює вміст вказівника pattern в duplicate.

**Parameters**

|  |  |
| --- | --- |
| **duplicate** | вказівник на My\_Double. Не може бути пустим вказівником. |
| **pattern** | вказівник на My\_Double. Не може бути пустим вказівником. |

**mdb\_string.h File Reference**

Основні маніпуляції з рядком і перетворення рядка в число типу My\_Double.

#include "[**my\_double.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h_source.html)"  
#include "[**adduction.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\adduction_8h_source.html)"

|  |  |
| --- | --- |
| 1. Functions | |
| int | [**member**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h.html#a9ddb8d8741e2d498b1372335282ca00d) (char \*bank, char symbol, int Radix) |
|  | Функція перевіряє, чи є змінна symbol елементом масиву bank з індексом, не більшим Radix. |
|  | |
| int | [**pos**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h.html#ab4234e5aa4f75410301d475f4977e683) (char \*bank, char symbol, int size) |
|  | Пошук символа. Функція шукає символ symbol в масив символів bank і повертає його індекс в масиві. |
|  | |
| int | [**asc2dbl**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h.html#adca919d02719329e0507a81bbb2efc85) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*pDbl, const char \*str, int Radix) |
|  | Перетворення із строкового представлення в числове. Функція перетворює рядок символів str в число типу My\_Double в системі числення з основою Radix. |
|  | |
| int | [**dblasc2**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\mdb__string_8h.html#a512ac399bbabaa875aa785563dd173ea) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) example, char \*str, int size, int Radix) |
|  | Перетворення із числового представлення в строкове. Функція формує рядок із числа example типу My\_Double в системі числення з основою Radix і записує його в рядок str, але заповнюючи при цьому не більше, ніж size символів. |
|  | |

Function Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **int asc2dbl** | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **pDbl,** |
|  |  | **const char \*** | **str,** |
|  |  | **int** | **Radix** |
|  | **)** |  |  |

Перетворення із строкового представлення в числове. Функція перетворює рядок символів str в число типу My\_Double в системі числення з основою Radix.

**Parameters**

|  |  |
| --- | --- |
| **pDbl** | вказівник на число типу My\_Double. Не може бути пустим. |
| **str** | вказівник на рядок. Не може бути пустим. |
| **Radix** | основа системи числення. Має бути цілим числом від 2 до 16. |

**Returns**

кількість символів, яке функція використовувала для запису із рядка в число.

-1, якщо в рядку забагато символів.

-2, якщо в рядку зустрівся символ, що не використовується в даній системі числення; або більше двох крапок в числі; або перед мінусом є інші символи.

-3, якщо неправильні параметри.

|  |  |  |  |
| --- | --- | --- | --- |
| **int dblasc2** | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) | **example,** |
|  |  | **char \*** | **str,** |
|  |  | **int** | **size,** |
|  |  | **int** | **Radix** |
|  | **)** |  |  |

Перетворення із числового представлення в строкове. Функція формує рядок із числа example типу My\_Double в системі числення з основою Radix і записує його в рядок str, але заповнюючи при цьому не більше, ніж size символів.

**Parameters**

|  |  |
| --- | --- |
| **example** | число типу My\_Double. |
| **str** | вказівник на рядок. Не може бути пустим. |
| **size** | кількість символів, які будуть заповнені. |
| **Radix** | основа системи числення. Має бути цілим числом від 2 до 16. |

**Returns**

кількість символів, записаних в рядок без знаку "-" і десяткової крапки.

0, якщо розмір рядка size надто малий для того, щоб в нього записати число з заданою точністю.

-1, якщо неправильні параметри.

|  |  |  |  |
| --- | --- | --- | --- |
| **int member** | **(** | **char \*** | **bank,** |
|  |  | **char** | **symbol,** |
|  |  | **int** | **Radix** |
|  | **)** |  |  |

Функція перевіряє, чи є змінна symbol елементом масиву bank з індексом, не більшим Radix.

**Parameters**

|  |  |
| --- | --- |
| **bank** | масив символів. Не може бути рівним NULL. |
| **symbol** | символ. |
| **Radix** | індекс. Не може бути меншим нуля. |

**Returns**

1, якщо такий елемент знайдений.

0, якщо такий елемент не знайдений.

-1, якщо неправильно передані параметри.

|  |  |  |  |
| --- | --- | --- | --- |
| **int pos** | **(** | **char \*** | **bank,** |
|  |  | **char** | **symbol,** |
|  |  | **int** | **size** |
|  | **)** |  |  |

Пошук символа. Функція шукає символ symbol в масив символів bank і повертає його індекс в масиві.

**Parameters**

|  |  |
| --- | --- |
| **bank** | масив символів. Не може бути рівним NULL. |
| **symbol** | символ, який потрібно знайти. |
| **size** | розмір масиву. Не може бути меншим нуля. |

**Returns**

від 0 до size якщо елемент знайдений.

-1, якщо такий елемент не знайдений.

-2, якщо неправильно передані параметри.

**my\_double.h File Reference**

Представлення числа з плаваючою точкою в пам'яті.

#include <limits.h>

|  |  |
| --- | --- |
| 1. Data Structures | |
| struct | [**my\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\structmy___double.html) |
|  | Структура, що представляє число з плаваючою точкою [More...](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\structmy___double.html#details) |
|  | |

|  |  |
| --- | --- |
| 1. Typedefs | |
| typedef struct [**my\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\structmy___double.html) | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) |
|  | Структура, що представляє число з плаваючою точкою |
|  | |

|  |  |
| --- | --- |
| 1. Enumerations | |
| enum | { [**EXP\_MAX**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a06fc87d81c62e9abb8790b6e5713c55ba66ffa0390964b376ff419dc395e57793) = CHAR\_MAX, [**MANTISS\_MAX**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a06fc87d81c62e9abb8790b6e5713c55ba8e3c09ee86ce550e70496a899edd7a4d) = CHAR\_MAX } |
|  | Перелік, в якому визначені максимальне значення порядку і мантиси [More...](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a06fc87d81c62e9abb8790b6e5713c55b) |
|  | |

Typedef Documentation

|  |
| --- |
| **typedef struct**[**my\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\structmy___double.html)[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) |

Структура, що представляє число з плаваючою точкою

Enumeration Type Documentation

|  |
| --- |
| **anonymous enum** |

Перелік, в якому визначені максимальне значення порядку і мантиси

**Enumerator:**

|  |  |
| --- | --- |
| *EXP\_MAX* | найбільше значення порядку |
| *MANTISS\_MAX* | найбільше значення мантиси |

**operations.h File Reference**

Арифметичні операції над числами з плаваючою крапкою.

#include "[**my\_double.h**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h_source.html)"

|  |  |
| --- | --- |
| 1. Functions | |
| void | [**div**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html#a733168d0f6e5aa54cfd13a575070cf26) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*dividend, const [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*denominator, int \*h) |
|  | Ділення двох чисел з плаваючою крапкою. Функція ділить два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр. |
|  | |
| void | [**mult**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html#a54254a414b07af55037bbe32872dc796) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*A, const [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*B, int \*h) |
|  | Множення двох чисел з плаваючою крапкою. Функція перемножає два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр. |
|  | |
| void | [**sub**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html#ab95bd79a8cf626a8d7b27c784c5cb8cf) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*U, const [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*V, int \*h) |
|  | Віднімання двох чисел з плаваючою крапкою. Функція віднімає два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр. |
|  | |
| void | [**add**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html#adccd0859ec91e4fd12beae9acc332a22) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*A, const [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*B, int \*h) |
|  | Додавання двох чисел з плаваючою крапкою. Функція додає два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр. |
|  | |
| void | [**rest**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html#a0c4c84f5845d939d7765a7c1eda9962b) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*A, const [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) \*B) |
|  | Залишок від ділення. Функція знаходить залишок від ділення першого числа на друге. Тобто від A віднімаємо B, поки B не стане більшим A. Результат записуєть в перший параметр. |
|  | |
| int | [**quotient**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\operations_8h.html#a71be710164e2d9b311b049cbaeacedbf) ([**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) A, [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) B) |
|  | Цілочисельне ділення. Функція рахує, скільки разів B поміщається в A. |
|  | |

Function Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| **void add** | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **A,** |
|  |  | **const**[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **B,** |
|  |  | **int \*** | **h** |
|  | **)** |  |  |

Додавання двох чисел з плаваючою крапкою. Функція додає два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр.

**Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | перший доданок і сума. Не може бути пустим вказівником. | | | | |
| **B** | другий доданок. Не може бути пустим вказівником. | | | | |
| **h** | якщо дорівнює -1, то відбулося переповнення типу. | | | | |
| **void div** | | | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **dividend,** |
|  | | |  | **const**[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **denominator,** |
|  | | |  | **int \*** | **h** |
|  | | | **)** |  |  |

Ділення двох чисел з плаваючою крапкою. Функція ділить два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр.

**Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **dividend** | | | ділене і частка. Не може бути пустим вказівником. | | |
| **denominator** | | | дільник. Не може бути пустим вказівником. Мантиса не може дорівнювати нулю. | | |
| **h** | | | якщо дорівнює -1, то відбулося переповнення типу. | | |
| **void mult** | | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | | **A,** |
|  | |  | **const**[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | | **B,** |
|  | |  | **int \*** | | **h** |
|  | | **)** |  | |  |

Множення двох чисел з плаваючою крапкою. Функція перемножає два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр.

**Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | перший множник і добуток. Не може бути пустим вказівником. | | | | |
| **B** | другий множник. Не може бути пустим вказівником. | | | | |
| **h** | якщо дорівнює -1, то відбулося переповнення типу. | | | | |
| **int quotient** | | | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) | **A,** |
|  | | |  | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b) | **B** |
|  | | | **)** |  |  |

Цілочисельне ділення. Функція рахує, скільки разів B поміщається в A.

**Parameters**

|  |  |
| --- | --- |
| **A** | перше число. Не може бути пустим вказівником і рівним нулю. |
| **B** | друге число. Не може бути пустим вказівником і рівним нулю. |

**Returns**

позитивна частка від ділення.

-1, якщо неправильні параметри.

|  |  |  |  |
| --- | --- | --- | --- |
| **void rest** | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **A,** |
|  |  | **const**[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **B** |
|  | **)** |  |  |

Залишок від ділення. Функція знаходить залишок від ділення першого числа на друге. Тобто від A віднімаємо B, поки B не стане більшим A. Результат записуєть в перший параметр.

**Parameters**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | перше число і результат. Має бути додатнім. Не може бути пустим вказівником. | | | | |
| **B** | друге число. Має бути додатнім. Не може бути пустим вказівником. | | | | |
| **void sub** | | | **(** | [**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **U,** |
|  | | |  | **const**[**My\_Double**](file:///E:\SkyDrive\Documents\Science\System_programming\5sem\course%20work\Floating_Point_Operations_For_Integer_Processors\FLIP\docs\html\my__double_8h.html#a21acf7368653087fb0bd34b48176100b)**\*** | **V,** |
|  | | |  | **int \*** | **h** |
|  | | | **)** |  |  |

Віднімання двох чисел з плаваючою крапкою. Функція віднімає два числа типу My\_Double як числа з плаваючою крапкою. Результат записуєть в перший параметр.

**Parameters**

|  |  |
| --- | --- |
| **U** | зменшуване і різниця. Не може бути пустим вказівником. |
| **V** | від'ємник. Не може бути пустим вказівником. |
| **h** | якщо дорівнює -1, то відбулося переповнення типу. |

Висновки

Результатом виконання роботи є бібліотека функцій, що дозволяє виконувати обчислення над числами з плаваючою комою. Реалізовано арифметичні операції, увід із рядка і перетворення в строкову форму. Тип чисел з плаваючою комою реалізований на базі вбудованих в мову C типів int і char.

Отримано навички роботи з Visual Studio 2012 та інструментом модульного тестування MSTest.

Список інформаційних джерел

Correctly Rounded mathematical library [З мережі] // LipForge. - http://lipforge.ens-lyon.fr/www/crlibm/index.html.

**Guillaume Revy Claude-Pierre Jeannerod** Floating-point Library for Integer Processors [З мережі] // GForge. - http://flip.gforge.inria.fr/.

**Hauser J.** Test Float [З мережі]. - http://www.jhauser.us/arithmetic/TestFloat.html.

**Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В.** Прикладна теорія цифрових автоматів [Книга]. - К. : Вид-во Нац. авіац. ун-ту "НАУ-друк", 2009. - сс. 213-282. - ISBN 978-966-598-608-9.

**Корнейчук В.И, Тарасенко В.П.** Основы компьютерной арифметики [Книга]. - К. : Корнійчук, 2003. - с. 176.

**М.А Карцев** Арифметика цифровых машин [Книга]. - М. : Наука, 1969. - с. 576.

Додаток А. Програмний код роботи

* 1. flipunittest.cpp

/\*\*

\* @file flipunittest.cpp

\* @brief Тестування бібліотеки "FLIP" за допомогою <b>MSTest</b>.

\*/

#include "CppUnitTest.h"

#include "..\FLIP\my\_double.h"

#include "..\FLIP\mdb\_manipule.h"

#include "..\FLIP\mdb\_string.h"

#include "..\FLIP\operations.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTestFLIP

{

int OVER\_FLOW = -1,

WRONG\_SYMBOL = -2,

WRONG\_PARAMETER = -3;

/\*\*

\* @brief Тестує функцію для переведення рядка символів в число типу <code>My\_Double</code>.

\*/

TEST\_CLASS(UnitTestStringToMyDouble)

{

My\_Double rabbit;

public:

/\*\*

\* @brief Перевіряється запис рядка "1.0" в число типу <code>My\_Double</code>.

\*/

TEST\_METHOD(TestWriteString)

{

char str[4] = "1.0";

Assert::AreEqual(3, asc2dbl(&rabbit, str, 10),

L"TestWriteString failed");

}

/\*\*

\* @brief Переповнення типу при вводі надто великого рядка.

\*/

TEST\_METHOD(TestTypeOverflow)

{

char \* superstr ="1000000000000000000000000000000000000000\

\0000000000000000000000000000000000000000";

Assert::AreNotEqual(OVER\_FLOW, asc2dbl(&rabbit, superstr, 10),

L"TestTypeOverflow failed");

}

/\*\*

\* @brief Спроба записати рядок із недопустимими для даної системи числення символами.

\*/

TEST\_METHOD(TestWrongSymbol)

{

char str[8] = "19443.1";

Assert::AreEqual(WRONG\_SYMBOL, asc2dbl(&rabbit, str, 9),

L"TestWrongSymbol failed");

}

/\*\*

\* @brief Спроба записати число з двоми десятковими точками.

\*/

TEST\_METHOD(TestDoublePoints)

{

char str[6] = "1.1.1";

Assert::AreEqual(WRONG\_SYMBOL, asc2dbl(&rabbit, str, 9),

L"TestDoublePoints failed");

}

/\*\*

\* @brief Мінус в неправильній позиції.

\*/

TEST\_METHOD(TestMinusPosition)

{

char str[6] = "5-322";

Assert::AreEqual(WRONG\_SYMBOL, asc2dbl(&rabbit, str, 6),

L"TestMinusPosition failed");

}

/\*\*

\* @brief У числа два мінуса.

\*/

TEST\_METHOD(TestDoubleMinus)

{

char str[9] = "-3-60-56";

Assert::AreEqual(WRONG\_SYMBOL, asc2dbl(&rabbit, str, 9),

L"TestDoubleMinus failed");

}

/\*\*

\* @brief Неправильне поєднання символів.

\*/

TEST\_METHOD(TestSymbolsCombination)

{

char str[10] = ".-3218444";

Assert::AreEqual(WRONG\_SYMBOL, asc2dbl(&rabbit, str, 11),

L"TestSymbolsCombination failed");

}

/\*\*

\* @brief Невірні символи в рядку.

\*/

TEST\_METHOD(TestInvalidSymbols)

{

char str[7] = "6 86r;";

Assert::AreEqual(WRONG\_SYMBOL, asc2dbl(&rabbit, str, 16),

L"TestInvalidSymbols failed");

}

/\*\*

\* @brief Основа системи числення менша, ніж два.

\*/

TEST\_METHOD(TestNumeralSystem1)

{

char str[5] = "0000";

Assert::AreEqual(WRONG\_PARAMETER, asc2dbl(&rabbit, str, 1),

L"TestNumeralSystem1 failed");

}

/\*\*

\* @brief Основа системи числення більша, ніж шістнадцять.

\*/

TEST\_METHOD(TestNumeralSystem17)

{

char str[10] = "1f3dae4ag";

Assert::AreEqual(WRONG\_PARAMETER, asc2dbl(&rabbit, str, 17),

L"TestNumeralSystem17 failed");

}

/\*\*

\* @brief Нульовий вказівник в якості числа.

\*/

TEST\_METHOD(TestNullPointerString)

{

char str[10] = "1";

Assert::AreEqual(WRONG\_PARAMETER, asc2dbl(0, str, 12),

L"TestNullPointer failed");

}

/\*\*

\* @brief Нульовий вказівник в якості рядка.

\*/

TEST\_METHOD(TestNullString)

{

Assert::AreEqual(WRONG\_PARAMETER, asc2dbl(&rabbit, 0, 12),

L"TestNullString failed");

}

};

/\*\*

\* @brief Тестує операцію додавання

\*/

TEST\_CLASS(UnitTestAdding)

{

public:

My\_Double A, B, C;

/\*\*

\* @brief Перевірка операції 1+(-1).

\*/

TEST\_METHOD(TestAddOneMinusOne)

{

int h;

init(1, &A);

init(-1, &B);

add(&A, &B, &h);

Assert::AreNotEqual('0', A.mantiss, L"TestAddOneMinusOne failed");

}

/\*\*

\* @brief Перевірка суми 1+(-0.001).

\*/

TEST\_METHOD(TestAddOnePlusMinusMillesimal)

{

int h;

init(1, &A);

B.mantiss = -100;

B.exp = -3;

C.mantiss = 999;

C.exp = -3;

add(&A, &B, &h);

Assert::IsFalse(dblcmp(A, C) == 0, L"TestAddOnePlusMinusMillesimal failed");

}

/\*\*

\* @brief Переповнення мантиси зверху INT\_MAX\*2^10 + INT\_MAX\*2^10.

\*/

TEST\_METHOD(TestAddOverflowMantissAbove)

{

int h;

A.exp = 10; //М\*2^10 + М\*2^10 = М\*2^11

A.mantiss = MANTISS\_MAX;

B.exp = 10;

B.mantiss = MANTISS\_MAX;

C.exp = 11;

C.mantiss = MANTISS\_MAX;

add(&A, &B, &h);

Assert::IsFalse(dblcmp(A, C) == 0, L"TestAddOverflowMantissAbove failed");

}

/\*\*

\* @brief Переповнення мантиси знизу -INT\_MAX\*2^2 - INT\_MAX\*2^2.

\*/

TEST\_METHOD(TestAddOverflowMantissBelow)

{

int h;

A.exp = 2; //-M\*2^2 -М\*2^2 = -М\*2^3

A.mantiss = -MANTISS\_MAX;

B.exp = 2;

B.mantiss = -MANTISS\_MAX;

C.exp = 3;

C.mantiss = -MANTISS\_MAX;

add(&A, &B, &h);

Assert::IsFalse(dblcmp(A, C) == 0, L"TestAddOverflowMantissBelow failed");

}

/\*\*

\* @brief INT\_MAX\*2 - INT\_MAX/2.

\*/

TEST\_METHOD(TestAddMantissDivTwo)

{

int h;

A.exp = 1;

A.mantiss = MANTISS\_MAX;

B.exp = -1;

B.mantiss = MANTISS\_MAX;

C.exp = 2;

C.mantiss = 5 \* (INT\_MAX / 8);

add(&A, &B, &h);

Assert::IsFalse(dblcmp(A, C) == 0, L"TestAddMantissDivTwo failed");

}

/\*\*

\* @brief Переповнення порядку зверху INT\_MAX\*2^INT\_MAX + 1.

\*/

TEST\_METHOD(TestAddOverflowExpAbove)

{

int h;

A.exp = EXP\_MAX; //надвелике число, до якого

A.mantiss = MANTISS\_MAX; //не можна нічого додати

B.exp = -EXP\_MAX;

B.mantiss = MANTISS\_MAX;

C.exp = EXP\_MAX;

C.mantiss = MANTISS\_MAX;

add(&A, &B, &h);

Assert::IsFalse(A.exp != C.exp || A.mantiss != C.mantiss,

L"TestAddOverflowExpAbove failed");

}

/\*\*

\* @brief Переповнення порядку знизу -INT\_MAX\*2^INT\_MAX-1.

\*/

TEST\_METHOD(TestAddOverflowExpBelow)

{

int h;

A.exp = EXP\_MAX; //число, від якого нічого

A.mantiss = -MANTISS\_MAX; //не можна відняти

B.exp = EXP\_MAX;

B.mantiss = -MANTISS\_MAX;

C.exp = EXP\_MAX;

C.mantiss = -MANTISS\_MAX;

add(&A, &B, &h);

Assert::IsFalse(A.exp != C.exp || A.mantiss != C.mantiss,

L"TestAddOverflowExpBelow failed");

}

/\*\*

\* @brief Запис в нульовий вказівник.

\*/

TEST\_METHOD(TestAddNullPointer)

{

int h;

My\_Double \* A1 = NULL;

My\_Double \* B1 = NULL;

add(A1, B1, &h);

Assert::IsFalse(A1 != NULL, L"TestAddNullPointer failed");

}

};

/\*\*

\* @brief Тестує операцію ділення.

\*/

TEST\_CLASS(TestDivision)

{

public:

My\_Double A, B, C;

/\*\*

\* @brief Перевірка операції 1/2 = 0.5 = 1\*2^(-1).

\*/

TEST\_METHOD(TestSimpleDivision)

{

int h;

A.exp = 0;

A.mantiss = 1;

B.exp = 0;

B.mantiss = 2;

C.exp = -1;

C.mantiss = 1;

div(&A, &B, &h);

Assert::IsFalse(dblcmp(A, C) != 0, L"TestSimpleDivision failed");

}

/\*\*

\* @brief Переповнення мантиси INT\_MAX\*2 / (INT\_MAX/2) = 4.

\*/

TEST\_METHOD(TestDivOverflowMantiss)

{

int h;

A.exp = 1; //M\*2/(M/2) = M\*2^2

A.mantiss = MANTISS\_MAX;

B.exp = -1;

B.mantiss = MANTISS\_MAX;

C.exp = 2;

C.mantiss = 1;

div (&A, &B, &h);

Assert::IsFalse((A.exp != C.exp || A.mantiss != C.mantiss),

L"TestDivOverflowMantiss failed");

}

/\*\*

\* @brief Переповнення порядку зверху (INT\_MAX\*2^INT\_MAX)/(1/2).

\*/

TEST\_METHOD(TestDivOverflowExpAbove)

{

int h;

A.exp = EXP\_MAX;

A.mantiss = MANTISS\_MAX;

B.exp = -1;

B.mantiss = -1;

div(&A, &B, &h);

Assert::IsFalse((A.mantiss != MANTISS\_MAX || A.exp != EXP\_MAX),

L"TestDivOverflowExpAbove failed");

}

/\*\*

\* @brief Переповнення порядку знизу 2^(-INT\_MAX)/2.

\*/

TEST\_METHOD(TestDivOverflowExpBelow)

{

int h;

A.exp = -EXP\_MAX;

A.mantiss = 1;

B.exp = 1;

B.mantiss = 1;

div(&A, &B, &h);

Assert::IsFalse((A.mantiss != 1 || A.exp != -EXP\_MAX),

L"TestDivOverflowExpBelow failed");

}

/\*\*

\* @brief Ділення на нуль.

\*/

TEST\_METHOD(TestDivByZero)

{

int h;

A.exp = 1;

A.mantiss = 12;

B.exp = 1;

B.mantiss = 0;

div(&A, &B, &h);

Assert::IsFalse((A.mantiss != 12 || A.exp != 1),

L"TestDivByZero failed");

}

/\*\*

\* @brief Запис в нульовий вказівник.

\*/

TEST\_METHOD(TestDivNullPointer)

{

int h;

My\_Double \* A1 = NULL;

My\_Double \* B1 = NULL;

div(A1, B1, &h);

Assert::IsFalse(A1 != NULL || B1 != NULL, L"TestDivNullPointer failed");

}

};

/\*\*

\* @brief Тестує операцію множення.

\*/

TEST\_CLASS(TestMultiply)

{

public:

My\_Double A, B, C;

/\*\*

\* @brief Перевірка операції -2\*24 = -48.

\*/

TEST\_METHOD(TestSimpleMultiply)

{

int h;

A.exp = 1;

A.mantiss = -1;

B.exp = 3;

B.mantiss = 3;

C.exp = 4;

C.mantiss = 3;

mult(&A, &B, &h);

Assert::IsFalse(!dblcmp(A, C), L"TestSimpleMultiply failed");

}

/\*\*

\* @brief Переповнення порядку зверху 2^INT\_MAX \* 2.

\*/

TEST\_METHOD(TestMulOverflowExpAbove)

{

int h;

A.exp = EXP\_MAX;

A.mantiss = MANTISS\_MAX;

B.exp = 1;

B.mantiss = 1;

C.exp = A.exp;

C.mantiss = A.mantiss;

mult(&A, &B, &h);

Assert::IsTrue((A.exp != C.exp || A.mantiss != C.mantiss),

L"TestMulOverflowExpAbove failed");

}

/\*\*

\* @brief Переповнення порядку знизу 2^(-INT\_MAX) \* 1/2.

\*/

TEST\_METHOD(TestMulOverflowExpBelow)

{

int h;

A.exp = -EXP\_MAX;

A.mantiss = 1;

B.exp = -1;

B.mantiss = 1;

C.exp = A.exp;

C.mantiss = A.mantiss;

mult(&A, &B, &h);

Assert::IsFalse((A.exp != C.exp || A.mantiss != C.mantiss),

L"TestMulOverflowExpBelow failed");

}

/\*\*

\* @brief Множення на нуль.

\*/

TEST\_METHOD(TestMulByZero)

{

int h;

A.exp = 2;

A.mantiss = 3;

B.exp = 1;

B.mantiss = 0;

mult(&A, &B, &h);

Assert::IsTrue(A.mantiss == 0,

L"TestMulByZero failed");

}

/\*\*

\* @brief Запис в нульовий вказівник.

\*/

TEST\_METHOD(TestMulNullPointer)

{

int h;

My\_Double \* A1 = NULL;

My\_Double \* B1 = NULL;

mult(A1, B1, &h);

Assert::IsFalse(A1 != NULL || B1 != NULL, L"TestMulNullPointer failed");

}

};

}

* 1. my\_double.h

#ifndef MY\_DOUBLE\_H\_

#define MY\_DOUBLE\_H\_

/\*\*

\* @file my\_double.h

\* @brief Представлення числа з плаваючою точкою в пам'яті.

\*/

#include <limits.h>

/\*\*

\* @brief Структура, що представляє число з плаваючою точкою

\*/

typedef struct my\_Double

{

char mantiss; //!< мантиса

char exp; //!< порядок

} My\_Double;

/\*\*

\* @brief Перелік, в якому визначені максимальне значення порядку і мантиси

\*/

enum {

EXP\_MAX = CHAR\_MAX, //!< найбільше значення порядку

MANTISS\_MAX = CHAR\_MAX //!< найбільше значення мантиси

};

#endif

* 1. adduction.h

#ifndef ADDUCTION\_H

#define ADDUCTION\_H

/\*\*

\* @file adduction.h

\* @brief Визначена функція для вирівнювання порядків двох чисел типу <code>My\_Double</code>.

\*/

#include "my\_double.h"

/\*\*

\* @brief Функція вирівнює порядки двох чисел.

\* Можлива втрата точності.

\* @param example вказівник на число типу <code>My\_Double</code>, до порядку якого

\* приводять. Не може бути пустим вказівником.

\* @param pattern вказівник на число типу <code>My\_Double</code>,

\* порядок якого треба зрівняти з example. Не може бути пустим вказівником.

\*/

void reduction(My\_Double\* example, My\_Double\* pattern);

#endif

* 1. adduction.c

/\*\*

\* @file adduction.cpp

\* @brief Код для @link adduction.h@endlink

\*/

#include "my\_double.h"

#include <stdio.h>

/\* ---------------------------------------------------------------- \*/

void reduction(My\_Double \* example, My\_Double \* pattern)

{

int stop = 0;

if (example == NULL || pattern == NULL)

return;

if (example->mantiss == 0)

example->exp = pattern->exp;

if(example->exp == pattern->exp)

return;

else

//поки не зрівняються порядки або поки не буде переповнення або втрата точності

while ((example->exp != pattern->exp) && !stop)

{

if (example->exp > pattern->exp)

{

if (((example->mantiss > 0)&&(example->mantiss < MANTISS\_MAX / 2)) || ((example->mantiss < 0) && (example->mantiss > -MANTISS\_MAX / 2)))

{

example->exp--;

example->mantiss \*= 2;

}

else

{

pattern->exp++;

pattern->mantiss /= 2;

}

}

else

{

if (((pattern->mantiss < MANTISS\_MAX / 2) && (pattern->mantiss > 0))

|| ((pattern->mantiss < 0) && (pattern->mantiss > -MANTISS\_MAX / 2)))

{

pattern->exp--;

pattern->mantiss \*= 2;

}

else

{

example->exp++;

example->mantiss /= 2;

}

}

}

}

* 1. mdb\_manipule.h

#ifndef MDB\_MANIPULE\_H

#define MDB\_MANIPULE\_H

/\*\*

\* @file mdb\_manipule.h

\* @brief Основні маніпуляції з числом типу <code>My\_Double</code>.

\*/

#include "my\_double.h"

/\*\*

\* @brief Функція ініалізує змінну <code>number</code> цілим числом <code>num</code>.

\* Можлива втрата точності.

\* @param num ціле число.

\* @param number змінна типу <code>My\_Double</code>. Не може бути пустим вказівником.

\*/

void init(int num, My\_Double \* number);

/\*\*

\* @brief Функція порівнює дві змінних типу <code>My\_Double</code>.

\* @param left змінна типу <code>My\_Double</code>.

\* @param right змінна типу <code>My\_Double</code>.

\* @return 0, якщо числа рівні.

\* @return 1, якщо <code>left</code> більше <code>right</code>.

\* @return -1, якщо <code>left</code> менше <code>right</code>.

\*/

int dblcmp(My\_Double left, My\_Double right);

/\*\*

\* @brief Функція копіює вміст вказівника <code>pattern</code> в <code>duplicate</code>.

\* @param duplicate вказівник на <code>My\_Double</code>. Не може бути пустим вказівником.

\* @param pattern вказівник на <code>My\_Double</code>. Не може бути пустим вказівником.

\*/

void mdbcpy(My\_Double \* duplicate, const My\_Double \* pattern);

#endif

* 1. mdb\_manipule.c

/\*\*

\* @file mdb\_manipule.cpp

\* @brief Код для @link mdb\_manipule.h@endlink

\*/

#include "my\_double.h"

#include "adduction.h"

#include "mdb\_string.h"

#include <stdio.h>

/\* ---------------------------------------------------------------- \*/

void init(int num, My\_Double \* number)

{

int i = 0;

if (number == NULL)

return;

if (num == 0)

{

number->mantiss = 0;

number->exp = 0;

return;

}

number->exp = 0;

//Якщо мантиса перевищує максимально допустиме значення,

//зменшуємо її, одночасно збільшуючи порядок

while (num > MANTISS\_MAX)

{

num = num / 2;

number->exp++;

}

while(num % 2 == 0) //Мантиса має бути непарним числом

{

num = num / 2; //Всі степені двійки заносяться в порядок

i++;

}

number->mantiss = num;

number->exp += i;

}

/\* ---------------------------------------------------------------- \*/

int dblcmp(My\_Double left, My\_Double right)

{

if (left.mantiss > 0 && right.mantiss < 0)

return 1;

if (left.mantiss < 0 && right.mantiss > 0)

return -1;

reduction(&left, &right);

if (left.mantiss == right.mantiss)

return 0;

else if (left.mantiss > right.mantiss)

return 1;

else return -1;

}

/\* ---------------------------------------------------------------- \*/

void mdbcpy(My\_Double \* duplicate, const My\_Double \* pattern)

{

if (pattern == NULL || duplicate == NULL)

return;

duplicate->exp = pattern->exp;

duplicate->mantiss = pattern->mantiss;

}

* 1. mdb\_string.h

#ifndef MDB\_STRING\_H

#define MDB\_STRING\_H

/\*\*

\* @file mdb\_string.h

\* @brief Основні маніпуляції з рядком і перетворення рядка в

\* число типу <code>My\_Double</code>.

\*/

#include "my\_double.h"

#include "adduction.h"

/\*\*

\* @brief Функція перевіряє, чи є змінна <code>symbol</code> елементом масиву

\* <code>bank</code> з індексом, не більшим <code>Radix</code>.

\* @param bank масив символів. Не може бути рівним <code>NULL</code>.

\* @param symbol символ.

\* @param Radix індекс. Не може бути меншим нуля.

\* @return 1, якщо такий елемент знайдений.

\* @return 0, якщо такий елемент не знайдений.

\* @return -1, якщо неправильно передані параметри.

\*/

int member(char \* bank, char symbol, int Radix);

/\*\*

\* @brief Пошук символа.

\* Функція шукає символ <code>symbol</code> в масив символів

\* <code>bank</code> і повертає його індекс в масиві.

\* @param bank масив символів. Не може бути рівним <code>NULL</code>.

\* @param symbol символ, який потрібно знайти.

\* @param size розмір масиву. Не може бути меншим нуля.

\* @return від 0 до <code>size</code> якщо елемент знайдений.

\* @return -1, якщо такий елемент не знайдений.

\* @return -2, якщо неправильно передані параметри.

\*/

int pos(char \* bank, char symbol, int size);

/\*\*

\* @brief Перетворення із строкового представлення в числове.

\* Функція перетворює рядок символів <code>str</code> в число типу

\* <code>My\_Double</code> в системі числення з основою <code>Radix</code>.

\* @param pDbl вказівник на число типу <code>My\_Double</code>. Не може бути пустим.

\* @param str вказівник на рядок. Не може бути пустим.

\* @param Radix основа системи числення. Має бути цілим числом від 2 до 16.

\* @return кількість символів, яке функція використовувала для запису із рядка в число.

\* @return -1, якщо в рядку забагато символів.

\* @return -2, якщо в рядку зустрівся символ, що не використовується в даній

\* системі числення; або більше двох крапок в числі; або перед мінусом є інші символи.

\* @return -3, якщо неправильні параметри.

\*/

int asc2dbl(My\_Double \* pDbl, const char \* str, int Radix);

/\*\*

\* @brief Перетворення із числового представлення в строкове.

\* Функція формує рядок із числа <code>example</code> типу

\* <code>My\_Double</code> в системі числення з основою <code>Radix</code> і

\* записує його в рядок <code>str</code>, але заповнюючи при цьому не більше,

\* ніж <code>size</code> символів.

\* @param example число типу <code>My\_Double</code>.

\* @param str вказівник на рядок. Не може бути пустим.

\* @param size кількість символів, які будуть заповнені.

\* @param Radix основа системи числення. Має бути цілим числом від 2 до 16.

\* @return кількість символів, записаних в рядок без знаку "-" і десяткової крапки.

\* @return 0, якщо розмір рядка <code>size</code> надто малий для того, щоб в

\* нього записати число з заданою точністю.

\* @return -1, якщо неправильні параметри.

\*/

int dblasc2(My\_Double example, char \* str, int size, int Radix);

#endif

* 1. mdb\_string.c

/\*\*

\* @file mdb\_string.cpp

\* @brief Код для @link mdb\_string.h@endlink

\*/

#include "my\_double.h"

#include "adduction.h"

#include "operations.h"

#include "mdb\_manipule.h"

#include <stdlib.h>

/\* ---------------------------------------------------------------- \*/

int dblasc2(My\_Double example, char \* str, int size, int Radix)

{

//банк символів

char bank[] = {'0', '1', '2', '3', '4', '5', '6', '7',

'8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};

//максимально можлива основа числення

const int MAX\_NUMERATION = sizeof(bank);

//кількість цифр після коми

const int MAX\_FIDELITY = 3;

int i, j = 0, h = 1, length = 0, sign = 1;

My\_Double T, T1, D, Copy(example);

//не можна писати у NULL вказівник або система числення не від 2 до 16

if (str == 0 || Radix < 2 || Radix > MAX\_NUMERATION)

return -1;

init(1, &D);

init(1, &T);

init(Radix, &T1);

//зводимо роботу з від'ємними числами до роботи з додатніми

if (example.mantiss < 0)

{

sign = 0;

example.mantiss \*= -1;

Copy.mantiss \*= -1;

}

//підрахунок кількості розрядів в числі example (Copy)

while (dblcmp(Copy, D) == 1)

{

div(&Copy, &T1, &h);

length++;

}

if (dblcmp(Copy, D) == 0)

length++;

//якщо в заданий розмір рядка число не поміщається, то запис не відбудеться

if (length + 1 + MAX\_FIDELITY > size)

return 0;

i = 0;

//отримання основи максимального розряду нашого числа

while (i < length - 1)

{

mult(&T, &T1, &h);

i++;

}

while (i >= 0)

{

//обчислюємо цифру поточного розряду

h = quotient(example, T);

//опускаємось на розряд нижче

rest(&example, &T);

//якщо символ цифри міститься в банку символів

if (h >= 0 && h < MAX\_NUMERATION)

{

//записуємо його в рядок

str[j] = bank[h];

//готуємось записувати наступний символ

j++;

}

//зменшуємо основу, оскільки перешли до нового розряду

div(&T, &T1, &h);

i--;

}

//ціла частина закінчилась

str[j] = '.';

j++;

//працюємо з дробовою частиною

while (i > -MAX\_FIDELITY-1)

{

h = quotient(example, T);

rest(&example, &T);

if (h >= 0 && h < MAX\_NUMERATION)

{

str[j] = bank[h];

j++;

}

div(&T, &T1, &h);

i--;

}

//все, запис закінчився

str[j] = '\0';

//в залежності від того, чи було число example від'ємним

//перезаписуємо рядок зі знаком мінус

if (sign == 0)

{

for (i = length + MAX\_FIDELITY + 1; i > 0; i--)

str[i] = str[i-1];

str[0] = '-';

str[length + MAX\_FIDELITY + 2] = '\0';

}

return length + (1 - sign) + 1 + MAX\_FIDELITY;

}

/\* ---------------------------------------------------------------- \*/

int member(char \* bank, char symbol, int Radix)

{

int i, sign = 0;

if (bank == NULL || Radix < 0)

return -1;

for (i = 0; i < Radix; i++)

if (\*(bank + i) == symbol)

sign = 1;

return sign;

}

/\* ---------------------------------------------------------------- \*/

int pos(char \* bank, char symbol, int size)

{

int i, position = -1;

if (bank == NULL || size < 0)

return -2;

for (i = 0; i < size; i++)

if (\*(bank + i) == symbol)

position = i;

return position;

}

/\* ---------------------------------------------------------------- \*/

int asc2dbl(My\_Double \* pDbl, const char \* str, int Radix)

{

char bank[] = {'0', '1', '2', '3', '4', '5', '6', '7',

'8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};

int length = 0, i = 0, minus = 0, point = 0, before\_point = 0, after\_point = 0;

int sign = 0, c, j;

int h = 1;

My\_Double T, D, T1, T2;

//якщо неправильні параметри

if (pDbl == 0 || str == 0 || Radix < 2 || Radix > 16)

return -3;

while (\*(str + length) != '\0')

{

//якщо знак мінус розташований не там

if (str[length] == '-' && length != 0)

return -2;

//якщо символи не ті або їх занадто багато

if (!member(bank, str[length], Radix) && str[length] != '.'

&& str[length] != '-' || point >= 2 || minus >= 2 || str[0] == '.')

return -2;

//підрахунок кількості десяткових крапок

if (str[length] == '.')

point++;

//підрахунок кількості мінусів

if (str[length] == '-')

minus++;

length++;

}

//якщо більше однієї точки

if (point >= 2)

return -2;

//підрахунок кількості цифр до і після коми

while (before\_point < length)

{

if (\*(str + before\_point) == '.')

break;

before\_point++;

}

//якщо в запис входить крапка або навпаки не входить,

//формули для обчислення різні

if (point)

after\_point = length - before\_point - 1;

else

after\_point = length - before\_point;

//не рахуємо "-" за розряд

if (\*str == '-')

before\_point--;

init(0, pDbl);

init(1, &D);

//Т - основа поточного розряду (1 або 10 або 100 і т.д.)

init(1, &T);

//Т2 - записуємо сюди одиниці поточного розряду (від 0 до 9)

init(1, &T2);

//Т1 - основа системи числення

init(Radix, &T1);

//якщо перший символ в рядку не '-', рахуємо розряди з нього

if (\*str != '-')

i = 0;

else

{

i = 1;

sign = 1;

}

//підрахуємо основу максимального розряду (одиниці, десятки, сотні тисяч, ...)

for (j = 0; j < before\_point - i - 1 + sign; j++)

mult(&T, &T1, &h);

//поки не дійдемо до кінця рядку

while (\*(str + i) != '\0')

{

//якщо символ не є крапкою

if (\*(str + i) != '.')

{

//копіюємо основу поточного розряду

mdbcpy(&D, &T);

//обчислюємо одиниці поточного розряду

c = pos(bank, \*(str + i), sizeof(bank));

//представляємо одиниці поточного розряду в типі My\_Double

init(c, &T2);

//перемножаємо одиниці на основу поточного розряду

//(вийде, наприклад, 3000 або 2\*9^4)

mult(&D, &T2, &h);

//добавляємо отриманий розряд в тип My\_Double

add(pDbl, &D, &h);

//якщо переповнення, то переходимо до наступного розряду

if (h == -1)

return h;

div(&T, &T1, &h);

}

i++;

}

//в залежності від знаку, перемножаємо отримане число на -1

if (sign)

pDbl->mantiss \*= -1;

//кількість використаних символів

return length;

}

* 1. operations.h

#ifndef MUL\_DIV\_H\_

#define MUL\_DIV\_H\_

/\*\*

\* @file operations.h

\* @brief Арифметичні операції над числами з плаваючою крапкою.

\*/

#include "my\_double.h"

/\*\*

\* @brief Ділення двох чисел з плаваючою крапкою.

\* Функція ділить два числа типу <code>My\_Double</code> як

\* числа з плаваючою крапкою. Результат записуєть в перший параметр.

\* @param dividend ділене і частка. Не може бути пустим вказівником.

\* @param denominator дільник. Не може бути пустим вказівником.

\* Мантиса не може дорівнювати нулю.

\* @param h якщо дорівнює -1, то відбулося переповнення типу.

\*/

void div(My\_Double \* dividend, const My\_Double \* denominator, int \* h);

/\*\*

\* @brief Множення двох чисел з плаваючою крапкою.

\* Функція перемножає два числа типу <code>My\_Double</code> як

\* числа з плаваючою крапкою. Результат записуєть в перший параметр.

\* @param A перший множник і добуток. Не може бути пустим вказівником.

\* @param B другий множник. Не може бути пустим вказівником.

\* @param h якщо дорівнює -1, то відбулося переповнення типу.

\*/

void mult(My\_Double \* A, const My\_Double \* B, int \* h);

/\*\*

\* @brief Віднімання двох чисел з плаваючою крапкою.

\* Функція віднімає два числа типу <code>My\_Double</code> як

\* числа з плаваючою крапкою. Результат записуєть в перший параметр.

\* @param U зменшуване і різниця. Не може бути пустим вказівником.

\* @param V від'ємник. Не може бути пустим вказівником.

\* @param h якщо дорівнює -1, то відбулося переповнення типу.

\*/

void sub(My\_Double \* U, const My\_Double \* V, int \* h);

/\*\*

\* @brief Додавання двох чисел з плаваючою крапкою.

\* Функція додає два числа типу <code>My\_Double</code> як

\* числа з плаваючою крапкою. Результат записуєть в перший параметр.

\* @param A перший доданок і сума. Не може бути пустим вказівником.

\* @param B другий доданок. Не може бути пустим вказівником.

\* @param h якщо дорівнює -1, то відбулося переповнення типу.

\*/

void add(My\_Double \* A, const My\_Double \* B, int \* h);

/\*\*

\* @brief Залишок від ділення.

\* Функція знаходить залишок від ділення першого числа на друге.

\* Тобто від <code>A</code> віднімаємо <code>B</code>, поки <code>B</code>

\* не стане більшим <code>A</code>. Результат записуєть в перший параметр.

\* @param A перше число і результат. Має бути додатнім. Не може бути пустим вказівником.

\* @param B друге число. Має бути додатнім. Не може бути пустим вказівником.

\*/

void rest(My\_Double \* A, const My\_Double \* B);

/\*\*

\* @brief Цілочисельне ділення.

\* Функція рахує, скільки разів <code>B</code> поміщається в <code>A</code>.

\* @param A перше число. Не може бути пустим вказівником і рівним нулю.

\* @param B друге число. Не може бути пустим вказівником і рівним нулю.

\* @return позитивна частка від ділення.

\* @return -1, якщо неправильні параметри.

\*/

int quotient(My\_Double A, My\_Double B);

#endif

* 1. operations.c

/\*\*

\* @file operations.cpp

\* @brief Код для @link operations.h@endlink

\*/

#include <stdlib.h>

#include "my\_double.h"

#include "adduction.h"

#include "mdb\_manipule.h"

#include "mdb\_string.h"

/\* ---------------------------------------------------------------- \*/

void div(My\_Double \* A, const My\_Double \* B, int \* h)

{

//спроба ділення на нуль чи запису в нульовий вказівник

if (A == 0 || B == 0 || B->mantiss == 0)

return;

My\_Double Bcopy(\*B);

while ((A->mantiss >= 0 && A->mantiss <= MANTISS\_MAX / 2)

|| (A->mantiss < 0 && A->mantiss <= -MANTISS\_MAX / 2))

{

//приводимо ділене до максимально допустимої мантиси, щоб не

//було втрати точності при діленні

if (A->exp != -EXP\_MAX)

{

A->exp--;

A->mantiss \*= 2;

}

else break;

}

while (A->mantiss / Bcopy.mantiss == 0)

{

if (Bcopy.exp != EXP\_MAX && Bcopy.mantiss / 2 != 0)

{

Bcopy.exp++;

Bcopy.mantiss /= 2;

}

else break;

}

//можливе переповнення зверху при відніманні

//від'ємного числа із позитивного

if (A->exp >= 0 && Bcopy.exp < 0)

if (A->exp < EXP\_MAX + Bcopy.exp)

{

A->exp -= Bcopy.exp;

A->mantiss /= Bcopy.mantiss;

}

else

{

(\*h) = -1;

return;

}

//можливе переповнення знизу при відніманні

//від'ємного числа із позитивного

else if (A->exp < 0 && Bcopy.exp >= 0)

if (A->exp > Bcopy.exp - EXP\_MAX)

{

A->exp -= Bcopy.exp;

A->mantiss /= Bcopy.mantiss;

}

else {(\*h) = -1;return;}

//при відніманні однознакових чисел переповнення малоймовірне

else

{

A->exp -=Bcopy.exp;

A->mantiss /= Bcopy.mantiss;

}

}

/\* ---------------------------------------------------------------- \*/

void mult(My\_Double \* A, const My\_Double \* B, int \* h)

{

if (A == 0 || B == 0)

return;

My\_Double Bcopy(\*B);

if (Bcopy.mantiss == 0)

{

A->mantiss = 0;

return;

}

while (abs(A->mantiss) > MANTISS\_MAX / abs(Bcopy.mantiss))

{

if (A->exp != EXP\_MAX)

{

A->exp++;

A->mantiss /= 2;

}

else break;

}

if (A->exp > 0 && Bcopy.exp > 0)

{

if (A->exp < EXP\_MAX - Bcopy.exp)

{

A->mantiss \*= Bcopy.mantiss;

A->exp += Bcopy.exp;

}

else

{

while (abs(A->mantiss) <= MANTISS\_MAX / abs(Bcopy.mantiss))

{

A->mantiss \*= 2;

A->exp--;

}

if (A->exp < EXP\_MAX - Bcopy.exp)

{

A->exp += Bcopy.exp;

A->mantiss \*= Bcopy.mantiss;

}

else

{

\*h = -1;

return;

}

}

}

else if (A->exp < 0 && Bcopy.exp < 0)

{

if (A->exp > -EXP\_MAX - Bcopy.exp)

{

A->mantiss \*= Bcopy.mantiss;

A->exp += Bcopy.exp;

}

else

{

while (abs(A->mantiss) / 2 != 0)

{

A->exp++;

A->mantiss /= 2;

}

if (A->exp > -EXP\_MAX - Bcopy.exp)

{

A->exp += Bcopy.exp;

A->mantiss \*= Bcopy.mantiss;

}

else

{

(\*h) = -1;

return;

}

}

}

else

{

A->mantiss \*= Bcopy. mantiss;

A->exp += Bcopy.exp;

}

}

/\* ---------------------------------------------------------------- \*/

void add(My\_Double \* A, const My\_Double \* B, int \* h)

{

if (A == NULL || B == NULL)

return;

My\_Double Bcopy(\*B);

if (A->mantiss == 0)

A->exp = B->exp;

int quarter = 0;

reduction(A, &Bcopy);

//в якій координатній чверті знаходиться це число

if (A->mantiss >= 0)

{

if (Bcopy.mantiss >= 0)

//1-ий квадрант A >= 0, B >= 0

quarter = 1;

else

//2-ий квадрант A >= 0, B < 0

quarter = 2;

}

else

if (Bcopy.mantiss < 0)

//3-й квадрант A < 0, B < 0

quarter = 3;

else

//4-й квадрант A < 0, B >= 0

quarter = 4;

switch (quarter)

{

case 1:

//переповнення виникає через суму двох великих чисел

if (A->mantiss < MANTISS\_MAX - Bcopy.mantiss)

A->mantiss += Bcopy.mantiss;

else

{

if (A->exp == EXP\_MAX)

{

//якщо відбулось переповнення порядку

//встановлюємо флаг і припиняємо обчислення

\*h = -1;

return;

}

else

{

//збільшуємо порядок

A->exp++;

//зменшуємо мантису

A->mantiss /= 2;

//зменшуємо мантису другого доданка

Bcopy.mantiss /= 2;

//додаємо мантиси

A->mantiss += Bcopy.mantiss;

}

}

break;

case 2:

if (A->mantiss + Bcopy.mantiss < MANTISS\_MAX

&& A->mantiss + Bcopy.mantiss > -MANTISS\_MAX)

A->mantiss += Bcopy.mantiss;

else

{

if (A->exp == EXP\_MAX)

{

//якщо відбулось переповнення порядку

//встановлюємо флаг і припиняємо обчислення

\*h = -1;

return;

}

else

{

A->exp++;

A->mantiss /= 2;

Bcopy.mantiss /= 2;

A->mantiss += Bcopy.mantiss;

}

}

break;

case 3:

//переповнення знизу

if (A->mantiss > -MANTISS\_MAX - Bcopy.mantiss)

A->mantiss += Bcopy.mantiss;

else

{

if (A->exp != EXP\_MAX)

{

A->exp++;

A->mantiss /= 2;

Bcopy.mantiss /= 2;

A->mantiss += Bcopy.mantiss;

}

else

{

\*h = -1;

return;

}

}

break;

case 4:

if (A->mantiss + Bcopy.mantiss < MANTISS\_MAX

&& A->mantiss + Bcopy.mantiss > -MANTISS\_MAX)

A->mantiss += Bcopy.mantiss;

else

{

if (A->exp != EXP\_MAX)

{

A->exp++;

A->mantiss /= 2;

Bcopy.mantiss /= 2;

A->mantiss += Bcopy.mantiss;

}

else

{

\*h = -1;

return;

}

}

break;

default:

return;

}

}

/\* ---------------------------------------------------------------- \*/

void sub(My\_Double \* U, const My\_Double \* V, int \* h)

{

if (U == NULL || V == NULL)

return;

My\_Double Vcopy(\*V);

Vcopy.mantiss \*= -1;

add(U, &Vcopy, h);

}

/\* ---------------------------------------------------------------- \*/

void rest (My\_Double \* A, const My\_Double \* B)

{

int h = 1;

My\_Double Acopy(\*A);

if (A == NULL || B == NULL)

return;

//неіснує залишку від ділення на нуль

if (B->mantiss <= 0)

return;

//від'ємні числа можна тимчасово замінити позитивними

if (A->mantiss < 0 && B->mantiss > 0)

{

h = 0;

A->mantiss \*= -1;

}

//поки зменшуване менше від'ємника

while (dblcmp(\*A, \*B) == 1)

{

//копіюємо зменшуване

mdbcpy(&Acopy, A);

sub(A, B, &h);

//якщо зменшуване не змінилось, значить точність занадто мала і

//треба запобігти нескінченному циклу

if (dblcmp(Acopy, \*A) == 0)

break;

}

//якщо зменшуване зрівнялось з від'ємником, то залишок буде рівним нулю

if (dblcmp(\*A, \*B) == 0)

sub(A, B, &h);

//відновимо знак мінус

if (h == 0)

A->mantiss \*= -1;

}

/\* ---------------------------------------------------------------- \*/

int quotient(My\_Double A, My\_Double B)

{

int q = 0;

int h = 1;

int i = 0;

My\_Double Acopy(A);

if (B.mantiss == 0) {

return -1;}

if ((A.mantiss > 0 && B.mantiss < 0) || (A.mantiss < 0 && B.mantiss > 0))

{

return -1;

}

while (dblcmp(A, B) == 1)

{

mdbcpy(&Acopy, &A);

sub(&A, &B, &h);

if (dblcmp(Acopy, A) == 0)

break;

else q++;

}

if (h == -1)

return h;

h = 1;

if (dblcmp(A, B) == 0)

{

sub(&A, &B, &h);

q++;

}

if (h == -1)

return h;

return q;

}