Documentation

Author: Apăvăloaiei Alexandru, group 911

Problem statement

Design and implement an application that has algorithms:

- for performing fundamental arithmetic operations (addition, subtraction, multiplication by one digit and division by one digit) on positive integers within the specified base set {2, 3, ..., 10 and 16}.
- for conversion of natural numbers between two user-defined bases belonging to the set {2, 3, ..., 10 and 16}, using:
    - substitution method;
    - successive divisions;
    - rapid conversions(only with bases 2, 4, 8 or 16);
    - base 10 as an intermediary base.

The application must include a user-friendly menu that enables users to verify and execute each operation and conversion method independently.

Operations

Addition in base ‘p’

What does this function do?

- Performs addition in base p of two numbers a and b.
- First, the numbers are made of the same length by adding leading zeros to the shorter number.
- Then, the addition is performed digit by digit, from right to left.
- If the sum of two digits is greater than or equal to p, then the carry is 1 and the digit is the sum minus the base.
- Otherwise, the carry is 0 and the digit is the sum.

- If the carry is 1 after the last digit, then the carry is added to the result.

Parameters

- a → first number
- b → second number
- p → base in which the addition is performed

Return

> The result of the addition in base p of a and b

Pseudo-code

```
function addition_in_base_p(a: string, b: string, p: intiger)
    a, b = make_same_length(a, b)
    result = ""
    carry = 0

    for i from length(a) - 1 to 0 step -1:
        digit = char_to_digit(a[i]) + char_to_digit(b[i]) + c

        if digit >= p:
            carry = 1
            digit -= p
        else:
            carry = 0
                end_if/else

        result = digit_to_char(digit) + result
        end_for

    if carry == 1:
        result = "1" + result
        end_if

    return result
```

Subtraction in base ‘p’

What does this function do?

- Performs subtraction in base p of two numbers a and b.
- First, the numbers are made of the same length by adding leading zeros to the shorter number.
- Then, the subtraction is performed digit by digit, from right to left.
- If the difference of two digits is negative, then the borrow is 1 and the digit is the difference plus the base.
- Otherwise, the borrow is 0 and the digit is the difference.
- At the end, the leading zeros are removed.

Parameters

- a → first number
- b → second number
- p → base in which the addition is performed

Return

> The result of the subtraction in base p of a and b

Pseudo-code

```
function subtraction_in_base_p(a: string, b: string, p: intig
    a, b = make_same_length(a, b)

    if base_p_to_base_10(a, p) < base_p_to_base_10(b, p):
        ERROR("First number must be greater than or equal to
        end_if

    result = ""
    borrow = 0

    for i from length(a) - 1 to 0 step -1:
        digit = char_to_digit(a[i]) - char_to_digit(b[i]) - b
        if digit < 0:
            borrow = 1
            digit += p
        else:
            borrow = 0
                end_if/else

        result = digit_to_char(digit) + result

    result = remove_leading_zeros(result)
    return result
```

Multiplication by a single digit in base ‘p’

What does this function do?

- Performs multiplication in base p of a number a and a digit b.
- First, we check if the length of a is smaller than the length of b. If so, there is a chance that a is a digit and b is the number.
- Then, we check if the second parameter is a digit. If not, we raise an error.
- After that, we perform the multiplication digit by digit, from right to left.
- The operations are first performed in base 10, then the result is converted in base p.
- The multiplication is performed from right to left.
- The carry is the result of the division of the product of the digit of a and b by p.
- The digit is the remainder of the division of the product of the digit of a and b by p.
- At the end, if the carry is different from 0, then it is added to the result and the leading zeros are removed.

Parameters

- a → number to be multiplied

- b → the digit by which the multiplication is performed
- p → base in which the multiplication is performed

Return

> The result of the multiplication in base p of a and digit b

Pseudo-code

```
function multiplication_by_one_digit_in_base_p(a: string, b:
    if length(a) < length(b):
        # commutativity
        return multiplication_by_one_digit_in_base_p(b, a, p)
        end_if

    if length(b) != 1:
        ERROR("Multiplication only by one digit!")
        end_if

    result = ""
    carry = 0

    for i from length(a) - 1 to 0 step -1:
        digit = (base_p_to_base_10(a[i], p) * base_p_to_base_
        carry = (base_p_to_base_10(a[i], p) * base_p_to_base_
        result = digit_to_char(digit) + result
        end_for

    if carry != 0:
        result = digit_to_char(carry) + result
        end_if

    result = remove_leading_zeros(result)
    return result
```

Division by one digit in base ‘p’

What does this function do?

- Performs division in base p of a number a and a digit b.
- First, we check if the second parameter is a digit. If not, we raise an error.
- Then, we check if the second parameter is 0. If so, we raise an error because we cannot divide by 0.
- After that, we perform the algorithm for division like we would on paper.
- All operations are performed in base 10, then the result is converted in base p.
- The division is performed from left to right.

Parameters

- a → divident
- b → divisor (digit)
- p → base in which the division is performed

Return

> A tuple containing the quotient and the remainder of the division in base p of a and digit b

Pseudo-code

```
function division_by_one_digit_in_base_p(a: string, b: string
    digits = "0123456789ABCDEF"

    if length(b) != 1:
        raise ValueError("Division only by one digit!")
    end_if

    if b == "0":
        raise ValueError("ERROR: Division by zero!")
    end_if

    quotient = ""
    divisor = base_p_to_base_10(b, p)  # Divisor is converted
    remainder = ""

    for i from 1 to length(a):
        remainder = remainder + a[i]
        quotient = quotient + digits[base_p_to_base_10(remain
        remainder = digits[base_p_to_base_10(remainder, p) %
    end_for

    quotient = remove_leading_zeros(quotient)
    return (quotient, remainder)
```

Algorithms

Rapid conversions

What does this function do?

- Algorithm which converts numbers which are in bases which are powers of 2.
- In our case, the possible bases are 2, 4, 8, 16.
- We use base 2 as an intermediary base, just like we would on paper.
- First, we convert the number to base 2, then we convert it to the destination base.

Observation:

> We do not check if the bases are valid because the functions base_p_to_base_2() and base_2_to_base_p() do that when they are called.

Parameters

- number → number to be converted
- sb → source base; must be a power of 2
- db → destination base; must be a power of 2

Return

> The number converted to the destination base

Pseudo-code

```
function rapid_conversion(number: string, sb: intiger, db: in
    if sb == db:
        # number is already in the destination base
        return number
    end_if

    number = base_p_to_base_2(number, sb)
    result = base_2_to_base_p(number, db)
    return result
```

Conversion with intermediary base 10

What does this function do?

- Algorithm which converts numbers from any base to any base by using base 10 as an intermediary base.
- First, we convert the number to base 10, then we convert it to the destination base.

Parameters

- number → number to be converted
- sb → source base
- db → destination base

Return

> The number converted to the destination base

Pseudo-code

```
function conversion_with_intermediary_base_10(number: string,
    if sb == db:
        # number is already in the destination base
        return number
    end_if

    intermediary_base_10 = base_p_to_base_10(number, sb)
    result = base_10_to_base_p(intermediary_base_10, db)
    return result
```

Substitution method

What does this function do?

- Algorithm which converts numbers from a smaller base to a bigger base by using the substitution method.
- First, we create a list where we will store all the products which will be later added together to get the result.
- First for loop: We form in each element of that list the power of the source base by multiplying in the destination base.
- Second for loop: We multiply each power we formed in the first for loop with the corresponding digit from the number.
- Third for loop: We add all the products together to get the result in the destination base, because we performed all the operations in the destination base.

Observation:

> 💡 This method is used when the *source base* < *destination base*.

Parameters

- number → number to be converted
- sb → source base
- db → destination base

Return

> The number converted to the destination base

Pseudo-code

```
function substitution_method(number: string, sb: integer, db:
    list_of_products_in_dest_base = ["1"] * length(number)

    for i from 0 to length(number) - 1:
        power = length(number) - i - 1

        for p from 1 to power:
            list_of_products_in_dest_base[i] = multiplication_
        end_for

        list_of_products_in_dest_base[i] = multiplication_by_
    end_for

    result = "0"

    for i from 0 to length(number) - 1:
        result = addition_in_base_p(result, list_of_products_
    end_for

    return result
```

Successive divisions method

What does this function do?

- Algorithm which converts numbers from a bigger base to a smaller base by using the successive divisions method.
- First, we divide the number by the destination base and we store the remainder at the beginning of the result.
- Second, we divide the quotient by the destination base and we store the remainder at the beginning of the result.

- We repeat this process until the quotient is 0 and then the result will be all the remainders in the order we concatenated them. All operations are performed in intermediary base 10.

Observation:

> This method is used when *source base* > *destination base*.

Parameters

- number → number to be converted
- sb → source base
- db → destination base

Return

> The number converted to the destination base

Pseudo-code

```
function successive_divisions_method(number: string, sb: inte
    result = ""

    while number != "0":
        quotient, remainder = division_by_one_digit_in_base_p
        number = quotient
        result = remainder + result
    end_while

    return result
```