Optional homework – implementation operations + conversions

Computational Logic 2023

Documentation

Grancea Alexandru-Nicolae

Group 913

Problem statement:

Write a program that processes numbers in miscellaneous number bases. That must do operations like:

* Basics operations in a base p read from the input (additions, subtractions, multiplications, divisions)
* Conversion between two bases using different methods like(substitution method, successive divisions and multiplications , intermediary base and rapid conversion between bases that are power of two’s (2, 4, 8, 16))

The interaction between user and program will be realized by a menu which specifies very clear what options do you have:

1. Add two numbers in a base p.
2. Subtract two numbers in a base p. (the first number > the second number)
3. Multiply a number with a digit in a base p.
4. Divide a number with a digit in a base p.
5. Convert a number from base p to base d using substitution method (p < d).
6. Convert a number from base p to base d using successive divisions and multiplication method (p > d).
7. Convert a number from base p to base d using 10 as an intermediate base.
8. Rapid conversion between two numbers in bases that are power of two’s.

For the first 2 options the user needs to write the base where he wants to make the operations and the numbers that he wants to add/subtract.

For the options 3-4 the user needs to write the base, the number that he wants to multiply and the digit for multiply/divide.

For option 5-8 the user needs to write the initial base p, the number that he wants to convert and the destination base.

For every option the program will print the expected output.

A diagram of a diagram

Description automatically generated

Data type specifications:

For the input numbers I used string from c++ library <string>. This let me process the data easier and also let me to make operations on big numbers. To calculate the arithmetic operations and conversions I use array of int types because I need to make calculations.

For exemple: if I read two numbers: a = 1110(2), b = 1001(2), a and b strings, I will make the addition in a integer vector c. First I reverse the strings a, b such that : a = 0111(2), b = 1001(2).

After I take at every step from left to right the value on the position and convert it to int and add to c. Finally, I will take every value from c and convert it to string and reverse the string I obtain.

Pseudocode:

Operations:

1. Addition

Input a = [a0, a1, … an-1], b = [b0, b1, …., bm] and base p e [2, 3, 4, …10, 16] and a, b need to be a valid number in base p.

Add (a, b, p)

Reverse(a), Reverse(b)

Add nesemnificative 0 at the front of the numbers.

I = 0, carry = 0, c = [] int

While I < size(a) or I < size(b) or carry != 0

Carry = carry + int(a[i]) + int(b[i])

C[i] = carry % p

Carry = carry / p

I = I + 1

End while

Ans = “”

Convert c to string

Ans = reverse(c)

Return ans

End add

1. Subtraction

Input a = [a0, a1, … an-1], b = [b0, b1, …., bm] and base p e [2, 3, 4, …10, 16] and a, b need to be a valid number in base p and a > b.

Subt(a, b, p)

Reverse(a), Reverse(b)

Add nesemnificative 0 at the front of the numbers.

I = 0, barrow = 0, c = [] int

While I < size(a) or barrow != 0

C[i] = c[i] + int(a[i]) – int(b[i]) - barrow

If c[i] < 0

barrow = 1

else barrow = 0

endif

c[i] = c[i] + barrow \* p

I = I + 1

Eliminate the nesignifiant 0

Ans = “”

Convert c to string

Ans = reverse(c)

Return ans

End subt

1. Multiply

Input a = [a0, a1, … an-1], digit and base p e [2, 3, 4, …10, 16] and a, digit need to be a valid number in base p

Mul(a, digit, p)

Reverse(a),

Add nesemnificative 0 at the front of the number.

I = 0, carry = 0, c = [] int

While I < size(a) or carry != 0

Carry = carry + int(a[i]) + int(digit)

C[i] = carry % p

Carry = carry / p

I = I + 1

End while

Ans = “”

Convert c to string

Ans = reverse(c)

Return ans

End Mul

1. Division.

Input a = [a0, a1, … an-1], digit and base p e [2, 3, 4, …10, 16] and a, digit need to be a valid number in base p

Div(a, digit, p, rest)

Reverse(a),

Add nesemnificative 0 at the front of the number.

I =size(a) – 1, t = 0, c = [] int

While I >= 0

t = t \* p + int(a[i])

c[i] = t / int(digit)

t = t % digit

I = I + 1

End while

Ans = “”

Convert c to string

Ans = reverse(c)

Rest = string(t)

Return ans

End Div

Conversions

1. Substitution method:

Input a = [a0, a1, … an-1], the initial base p and the destination base dest. A needs to be a valid number in base p and also p < dest. A can be a real number.

Sub\_method(a, p, dest)

Poz = Find the position of the coma if a is a real number.

Ans\_int = “”

Power = int(p)

For I = poz – 1, 0, -1

Aux = Multiply(Power, a[i], dest)

Ans\_int = Add(ans\_int, x, dest)

Power = Multiply(Power, int(p), dest)

End for

If a is int

Return ans\_int

Else

Ans\_float = “”

For I = poz + 1, size(a), 1

Nrdiv = I – poz

While Nrdiv != 0

Aux = Divide(a[i], Power, dest, r)

X = aux

End while

Ans\_float = Add(ans\_float, aux, dest)

End for

Eliminate the insemnificative 0.

Return ans\_int+ans\_float

End if

1. Succesive divisions and multiplications method

Input a = [a0, a1, … an-1], the initial base p and the destination base dest. A needs to be a valid number in base p and also p > dest. A can be a real number.

Succ\_div\_mul(a, p, dest)

Poz = Find the position of the coma if a is a real number.

Ans\_int = “”

Int\_part = “”

For I = 0, poz, 1

Int\_part = int\_part + a[i]

End for

While size(int\_part) != 0

Int\_part = Division(int\_part, int(dest), p, rest)

Ans\_int = ans\_int + rest

End while

Reverse(int\_part)

If a is int

Return ans\_int

Else

Ans\_float = “”

Float\_part = “0”

For I = poz + 1, size(a), 1

Float\_part = float\_part + a[i]

End for

I = 0

While I < 10

Float\_part = Multiply(float\_part, digit, p)

Ans\_float = int(float\_part[0])

Float\_part[0] = ‘0’

I = I + 1

End while

Eliminate the insemnificative 0.

Return ans\_int+ans\_float

End if

1. Intermediare base convertor:

Input a = [a0, a1, … an-1], the initial base p and the destination base dest. A needs to be a valid number in base p. A can be a real number.

Intermediare\_base\_con(a, p, dest)

Ans = “”

If p > 10

Ans = Succesive\_div\_mult(a, p, 10)

Else ans = Substitution\_method(a, p, 10)

End if

If dest < 10

Ans = Succesive\_div\_mult(ans, 10, dest)

Else Ans = Substitution\_method(ans, 10, p)

Return ans

1. Rapid conversion

Input a = [a0, a1, … an-1], the initial base p and the destination base dest. A needs to be a valid number in base p and p and dest are powers of two . A can be a real number.

Rapid\_conv(a, p, dest)

Aux = “”

For I = 0, I < size(a), 1

If a[i] != ‘,’

X = Succesive\_div\_mult(a[i], p, 2)

Complete with nesemnificative 0 to the log(p)

Aux = aux + x

Else aux = aux + ‘,’

End if

End for

Ans = “”

Poz = Find the coma if a is a real number

Lg = log2(dest)

For I = poz – 1, lg – 1, I = I – lg

X = aux[I – lg + 1, lg]

X = Substitution\_method(x, 2, dest)

Ans = ans + x

End for

Eliminate insignificant 0

Reverse(ans)

If a is int

Return ans

Else

For I = poz + 1, I < size(aux) – lg, I = I + lg

X = aux[I, lg]

X = Substitution\_method(x, 2, dest)

Ans = ans + x

End for

Return ans

End if

Test data:

Addition:

1. Add\_numbers\_base\_p(54AB6F, CD097D, 16) = 121B4EC
2. Add\_numbers\_base\_p(23045, 100254, 6) = 123343
3. Add\_numbers\_base\_p(1100101011, 11101101, 2) = 10000011000

Subtrction:

1. Sub\_in\_base\_p(501BA, 32ED, 16) = 4CECD
2. Sub\_in\_base\_p(102387, 64502, 9) = 26785
3. Sub\_in\_base\_p(102003, 3333, 4) =32010

Multiplication:

1. Multiply\_in\_basep\_digit(A23F4, B, 16) = 6F8B7C
2. Multiply\_in\_basep\_digit(7023, 5, 8) = 43137
3. Multiply\_in\_basep\_digit(31203, 3, 5) = 144114

Division:

1. Division\_in\_basep\_digit(2A0F86, E, 16) = 3011B, remainder C
2. Division\_in\_basep\_digit(20101, 2, 3) = 10012, remainder 0
3. Division\_in\_basep\_digit(765433, 4, 8) = 175306, remainder 3

Substitution method:

1. Substitution\_method(3041,23, 5, 10) = 396,52
2. Substitution\_method(1735,62, 8, 10) = 989,78125
3. Substitution\_method(1556,23, 7, 16) = 275,58D0FA

Succesive divisions and multiplications method:

1. Succesive\_div\_mul\_basep(7048,56, 9, 4) = 1100123,220102
2. Succesive\_div\_mul\_basep(2510,43, 7, 3) = 1021202,122
3. Succesive\_div\_mul\_basep(2653,14, 10, 6) = 20141,05

Intermediare base:

1. Intermediare\_base\_converter(2510,43, 7, 3) = 1021202,122
2. Intermediare\_base\_converter(1735,62, 8, 10) = 989,78125
3. Intermediare\_base\_converter(7048,56, 9, 4) = 1100123,220102

Rapid conversion:

1. Rapid\_conversion(BC13F,57032, 16, 2) = 10111100000100111111,0101011100000011001
2. Rapid\_conversion(1230,321, 4, 8) = 154,71
3. Rapid\_conversion(63401,527, 8, 16) = 6701,CB8