JULPROJEKT 16/17

Konvertering talbaser

BILD

Projekt för kursen
Applicerad Yrkesmatematik

https://github.com/GoblinDynamiteer/XmasProject2016

Johan Kämpe
UIS16

2017-01-01

Noteringar

Jag har valt att namnge mitt konverteringsbibliotek till *libconvert.c* med headerfil *libconvert.h*

Jag har valt att använda egna namn på funktionerna i biblioteket. Men för att testerna i *tests.h* ska gå att använda utan att modifiera innehållet i koden, så har jag skapat tre extra funktioner i *libconvert.c:*

- char ***convert_to_binary**(int num)
- char ***convert_to_base**(int num, int base)
- char ***convert_to_base_frac**(double decimal, int maxDigits, int base)

Dessa tre funktioner använder mina egna konverteringsfunktioner för att ge ett konverterat returvärde. Det går antagligen att lösa detta på ett annat, snyggare sätt.

Funktioner i libconvert.c

Konvertering av decimala tal till godtycklig talbas:

Funktionerna ger en pekare till en char-array som returvärde.

convertIntDecToBase: konverterar ett decimalt heltal till angiven talbas.
convertFracDecToBase: konverterar decimalt tal mindre än 1 till angiven talbas.
convertDecToBase: Använder ovanstående två funktioner för att konvertera ett decimalt tal till angiven talbas. Det decimala talet kan vara ett heltal eller deltal. Exempel 10.34

Konvertering av tal med godtycklig talbas till decimala tal:

convertIntBaseToDec: konverterar heltal av angiven talbas till decimalt heltal, returtyp int.
convertFracBaseToDec: konverterar tal mindre än 1 av angiven talbas till decimalt tal, returtyp är double.
convertBaseToDec: Använder ovanstående två funktioner för att konvertera ett tal av angiven talbas. Det angivna talet kan vara ett heltal eller deltal. Exempel A35.34F, returtyp är double.

Konvertering mellan talbaser:

convertBaseToBase: Använder funktionerna *convertBaseToDec* och *convertDecToBase* för att konvertera ett tal från en angiven talbas till en annan.

Andra stödfunktioner:

reverseString: Vänder på tecknen i en char-array, denna funktion är identisk med den som kodades gemensamt i skolan.
numToChar: Omvandlar ett heltal till en bokstav, för talbaser större än 10.
charToNum: Omvandlar en bokstav till ett heltal.
stripZeroes: Tar bort nollor högerifrån i en char-array används för att t.ex. konvertera 0.23100 till 0.231.
powerOf: Eget alternativ till pow() från math.h. Används dock inte i nuläget.

Originalfilerna i talbas.zip

Jag har valt, förutom att använda mitt egna funktionsbibliotek libconvert, också valt att göra julprojektets uppgifter efter den färdigskrivna kodbasen från TomKi.

För övningens skull skrev jag kommentarer på engelska i **conv-to-dec.c** och **conv-to-base.c**

TomKis tester för konverteringsfunktionerna:

```
SUCCESS 826(10) = 1100111010(2) = 1100111010(2)
SUCCESS 128(10) = 10000000(2) = 10000000(2)
SUCCESS 255(10) = 11111111(2) = 11111111(2)
SUCCESS 0(10) = 0(2) = 0(2)
SUCCESS 826(10) = 1472(8) = 1472(8)
SUCCESS 365(10) = 555(8) = 555(8)
SUCCESS 128(10) = 200(8) = 200(8)
SUCCESS 0(10) = 0(8) = 0(8)
SUCCESS 826(10) = 33A(16) = 33A(16)
SUCCESS 365(10) = 16D(16) = 16D(16)
SUCCESS 128(10) = 80(16) = 80(16)
SUCCESS 0(10) = 0(16) = 0(16)
SUCCESS 0.375(10) = .011(2) = .011(2)
SUCCESS 0.1(10) = .0001100110(2) = .0001100110(2)
SUCCESS 0.1416(10) = .001001000011111(2) = .001001000011111(2)
SUCCESS 0(10) = .0(2) = .0(2)
SUCCESS 0.375(10) = .6(16) = .6(16)
SUCCESS 0.1(10) = .199999A(16) = .199999A(16)
SUCCESS 0.1416(10) = .243FE5C(16) = .243FE5C(16)
SUCCESS 0(10) = .0(16) = .0(16)
PI = 11.00100100001111110110(2)
PI = 3.243F6A8(16)
convert_to_base_both:
PI = 3.243F6A8(16)
```

```
11.00100100001111110110 (bas 2) = 3.14159
3.243F6A8 (bas 16) = 3.14159
```

Problem och funderingar

malloc

För funktioner som returnerar en pekare till en char-array använder jag *malloc* för att datan inte ska försvinna när funktionen är färdig. Dock frigör jag aldrig detta minnesutrymme, vilket jag har förstått inte är speciellt bra. Jag har provat att använda funktionen free() på lite olika ställen i koden men det gör att mina konverteringar inte blir korrekta, eller att skräptecken visas.

pow / typecasting

I funktionen **convertIntBaseToDec** som konverterar ett tal av angiven talbas till decimalt heltal har jag haft ett problem med avrundningar.

När jag använde kodraden nedan hände det ofta att talet blev 1 för litet.:

converted += **charToNum**(number[i]) * **pow**(base, powerOf);

Exempel
charToNum(number[i]) = 2
pow(10, 2)

$2 * 10^2 \neq 199$

Detta upptäcktes när jag gjorde tester för att konvertera ett decimalt tal till ett decimalt tal med min funktion **convertBaseToBase.**

Jag löste problemet genom att byta ut raden till:

converted += **charToNum**(number[i]) * (int)(**pow**(base, powerOf) + 0.5);

Nu fungerar det, men jag är väldigt osäker på om det är en bra/korrekt lösning. Finns det risk att talet nu blir 1 för stort i stället?

Jag har sett i Tomkis kod att en annan algoritm används:

converted *= base + **charToNum**(number[i]);

Denna hade löst mitt problem, och den använder inte ens någon potens-funktion, samt kör char-arrayen framlänges!

Detta innebär att den kan användas för att enklare bygga en enda funktion som konverterar ett flyttal från valfri bas till decimalt tal, och inte använda tre funktioner för samma sak, som jag gör för tillfället.

Jag har dock valt att inte använda denna algoritm än, för jag förstår inte hur den fungerar, matematikmässigt.