このプログラムは、29^400+2000C1000を計算し、その結果を正確に全桁表示するプログラムである。以下にその説明と考察を記す。

説明

[bignum\_main.c]

このファイルはプログラムの中心となる部分である。

多倍長整数はint型配列で確保する。今回の場合では、num1とnum2がそれにあたる。配列の要素数は多倍長整数の桁数にあたり、int型グローバル変数digitで指定されている。また、一桁あたりの上限の数をdecimalで指定されている。つまり、多倍長整数は”decimal進数で(digit+1)桁”の自然数と言う形で表現されている(配列の要素は0番目から始まるため、桁数はこれよりもdigitよりも1だけ多い)。例えば、配列のi番目の数字がaだとすると、10進数に直すと、a×decimal^(digit)を意味している。

今回、最終的な解が601桁の数値であるということが判明しているため、decimal=10000=10^4と設定すると、必要なdigitの最小値は、digit>601/log(10)(decimal)を満たす最小の自然数であることが分かる。

これを解くと、digit=151とすれば、解を誤差無く正確に表示できる。

decimalを10^4としたのは、int型の有効範囲に起因する。int型の範囲は、-2147483648~2147483647であるから、10^9<2147483647より、最大でもdecimalを10^9まで設定することができる。しかし、今回のプログラムでは、多倍長×整数の計算が必要となるため、整数同士を掛けた結果が変数の型の有効範囲をこえないようにしなければならない。ここで、decimal=10^4と設定すると、decimal^2=10^8<2147483647となり、int型の有効範囲に収まり、桁溢れしない。こういった事情から、今回掛け算を実行する関数の整数の引数もdecimalよりも小さくなるように仕様として制限している(今回の問題ではかける数の最大値は2000<decimal=10^4であるため、このような制限を施しても問題は生じない)。

main関数ではまず、29^400の答えを格納し、最終的な答えを格納するためのint型配列num1と2000C1000(以後、組合せのCombinationをC(2000,1000)というように表記する)の答えを格納するための配列num2を宣言する。

ヘッダーファイル”bignum\_funcs.h”に含まれるnumPower関数を用いて、29^400を計算し、答えをnum1に記録する。

確認のため、この答えをprintf関数と”bignum\_funcs.h”に含まれるprintnum関数で端末に表示する。

今度は、ヘッダーファイル”bignum\_funcs.h”に含まれるnumCombi関数を用いて、C(2000,1000)を計算し、答えをnum2に記録する。

同様に、確認のため、この答えをprintf関数と”bignum\_funcs.h”に含まれるprintnum関数で端末に表示する。

最後に”bignum\_funcs.h”に含まれるnumAdd関数を用いて、num1とnum2の数値を足し、答えをnum1に記録し、最終結果を端末に表示し、プログラムを終了する。

使用した関数の含まれる bignum\_funcs.hの元となっているファイル、 bignum\_funcs.cに関しては次の項目で詳しく説明する。

[bignum\_funcs.c]

このファイルは、 bignum\_main.cで使用する関数や、その関数を構成する上で必要な関数を定義し使用する補助的な役割を果たしている。

bignum\_main.cで設定した、int型グローバル変数digitとdecimalは引き続きこのファイル内でも使用する。

initialize0関数は、int型配列を引数にとり、配列の全ての要素の値を0に代入して初期化する関数である。

subNum4int関数は、int型配列numとint型変数aを引数にとり、numにaを代入する関数である。

まず、numの後ろの方の要素に数値が残っていては、正しい結果が得られないので、一つ前に宣言した関数、initialize0関数で0で初期化する。その後に、aをdecimalで割った余りをnum[0]に代入する。これは、numがdecimal進数で、num[0]が一の位を表すため、aのうちdecimal未満の部分を代入する方法としてこのような操作をしている。num[1]以降のnum[i]の値は、decimal^iの位を表すので、aを順にdecimalで割った商の値をdecimalの余りの分だけ格納していく。aもdecimalもともにint型であるから、a/decimalの値も整数までとなり、この値には余りの数値は影響しない。今回、aは正でint型であるから、最大でも2147483647であり、decimal^3=10^12>2147483647より、num[2]までの値に必ず収まる。

subNum4Num関数はint型配列num1とnum2を引数に取って、num2の値をnum1にコピーする関数である。

subNum4int関数と同様の理由で、まず、num1を0で初期化し、その後にnum2の全桁を一つ一つforループでnum1に代入していくといったものになっている。

printnum関数はint型配列numを引数に取って、numの中にある値を10進数として端末上に表示する関数である。

まず、この関数内でforループのためのループ変数iとchar型文字列(文字配列)変数formatstrを宣言する。このformatstrというのは、現在動的に変数として設定しているdecimalから、表示すべき桁数を計算し、文字列として格納するための変数である。次に、numの要素を添字の大きい方から順に探っていき、0でないものにあたったらその数値をまず一つだけ表示し、iを一つ減じてforループを抜ける。次に、sprintf関数でdecimalの10を底とする対数を取って、表示すべき桁数を指定して文字列に格納する。これは、最上位以下の桁を表示する上で、0を埋めて表示する必要があるために行う操作である。最後に配列要素の先頭までループで回して、直前で決定した文字列に従って表示していく。なお、最初のforループ内でif文によって条件分岐している理由は、このforループ内の表示の条件をnum[i]!=0のみにしてしまうと、numが0だった場合に何も表示されずにこの関数が終了してしまうからである。

numAdd関数はint型配列num1とnum2を引数に取って、num1とnum2の和をnum1に記録する関数である。

基本的には各桁を順にforループで足してnum1にその答えを格納していく手順で問題ないが、num1[i]とnum2[i]の値を足した値がdecimal以上となる際には繰り上がり処理をして、num1[i+1]にその繰り上がり分を足す必要がある。足す操作は、subNum4int関数の説明で記した通り、num[i]の余りをそのままnum[i]に格納、num[i]をdecimalで割った商をnum[i+1]に格納するといった手順になっている。今回は、余りの計算と代入の操作を同時に行うため、実際の手順としては、商を計算してi+1番目の要素に足すといった操作が先になっている。

万が一、繰り上がり処理をした時に桁数が足りなくなった場合には、足し算で桁溢れしたことを端末上に表示して、異常終了する。

numMultiple関数はint型配列numとint型変数aを引数に取り、numとaの積をnumに格納する関数である。

今回の問題では、負の数は扱わないため、万が一引数aが負であった場合はエラーとして、異常終了するような仕様としている。 また、aが0であった場合は、numはいかなる場合でも0になるため、initialize0関数でnumに0を代入する(この後の操作にはaが0であるため、一切影響しない)。bignum\_main.cの多倍長整数についての説明の最後にあった、「かける数をdecimal以下に制限する」といった仕様もここで実装している。

例外処理を一通り終えた後に、numの各桁とaを順に掛けていき、numのそれぞれの桁に代入する。この時、例外処理でaの値をdecimal以下に制限したため、各桁の掛けた後の結果の値が、int型の有効範囲を超えて桁溢れする可能性はない。掛け算の操作が終了した後、もう一度forループを回し、各桁がdecimal未満に収まっているか、確認する。収まっていない場合は、繰り上がり処理を実行する。桁数が足りなくなった場合は異常終了する。

numDevide関数はint型配列numとint型変数dを引数にとり、numをdで割った商をnumに格納、numをdで割った余りを戻り値として返す関数である。

まず、例外処理として、割り算で定義されていない0で数値を割る場合、すなわちd=0の場合、今回の問題では扱わないdが負の場合、dがdecimalを超える値の場合を検知して異常終了する。dがdecimalを超える場合には、dを一旦多倍長整数に格納してから割り算を実行する必要があるが、今回の場合の割る数は最大でも1000であり、decimalよりも小さいため、このような制限を行っても問題はない。

例外処理の後には、各桁をdで割っていき、最下位の桁以外ではその余りを一つ下の桁に繰り下げ、最下位の桁では余りを戻り値として返し、関数を終了する。

numPower関数は引数としてint型配列numとint型変数a、xを取り、aのx乗をnumに格納するものである。

まず、底a=0の場合、累乗x=1の場合はそれぞれ0と1を代入する(後の操作が答えには影響しない)。aやxが負の場合は今回の問題設定ではあり得ないので、端末上で異常を知らせて、終了する。

次に、numにaを代入する。これはaを1乗したことと等しい。その後、numMultiple関数とwhile文を用いてx-1回だけnumにaを掛けていく。

numCombi関数はint型配列numとint型変数nとrを引数として取り、n個の中からr個のものを選ぶ組合せの総数を計算し、numに答えを記録する関数である。

まず、nやrが負の場合やn<rの場合は組合せの定義に反するため、検出して異常終了する。

次に、n!/r!/(n-r)!を計算する(すなわち、1~nまでの数値を掛けたり割ったりしていく)ので、答えを格納する変数であるnumに1を代入して初期化しておく。

最後にforループでi=1~rまで計算する。ここで、i番目について考えると、numにn-i+1を掛け、iで割るという操作をしている。この時、iで割った際にnumが割り切れなかった場合には、異常終了する。

このようにすると、num=n×(n-1)×(n-2)×・・・×(n-r+1)/(1×2×・・・×r)=n!/{r!×(n-r)!}となり、確かに組合せの総数が計算できる。

ここでnumにはnを掛けて、1で割って・・・といったように、かける数は大きい方から、割る数は小さい方から順に計算している。この理由は、このような方法で計算すると、途中であってもnumは割り切れて必ず整数となり、誤差が生じないためである。自然数aについて考えると、numがaで割りきれるためには、numはaの倍数である必要がある。自然数を順に変化させた場合、aの倍数はa番おきに現れる。自然数aで割る前には、numはnから順に1ずつ減少しながらa個の自然数を掛けたものであるから、この中にはaの倍数が必ず含まれる。よってnumは必ずaで割り切れる。この事実は全ての自然数aに対して成立するので、numは常に割り切れて整数となる。

以上の事実をもとに、numをiで割った余りを計算のチェックに使用することができ、答えの精度により確実性が増す。

考察

多倍長整数の扱いと改善方法について考える。

今回は、扱う数値がまだ小さかったため、全てint型の範囲でプログラムを完成させることができたが、扱う数値がより大きくなると対応できなくなってしまう。具体的には掛け算の部分である。

掛け算の結果は最大でも掛けられる数と掛ける数の桁数の和の桁数が必要になる。従って、答えの配列の桁数はdigit×2だけ必要になる。この時にdecimalを大きく取りすぎると答えがint型では足りなくなり、繰り上がり処理が正確に行えないため、int型よりもデータ範囲の大きな変数の型を使う必要がある。例えば、掛け算の答えを一時的に保存しておく変数として、longlong型の変数を宣言しておくと、9223372036854775807までの数値を扱えるようになり、decimalは10^9まで扱うことができ、配列の桁数を少なくすることが出来る(現在のものよりも桁数を半分以下に押さえることが出来る)。

また、掛け算や割り算についても、多倍長整数同士の演算で定義すると、より柔軟に対応できる。アルゴリズムは多倍長×整数の場合と同様の方法で行い、結果を一時的にlonglong型で保存して繰り上がり処理を行えば実現できる。

小数についても1の位の格納されている添字位置を表す変数を別の変数で記録しておくことで、対応で切るものだと考えられる。

負の数については、例えば、最上位の桁を数値ではなく、0または1のみを代入できるようにし、0の場合は正、1の場合は負といったように設定すると多倍長の場合でも表示できると思われる。