

Kalkulator

Dokumentacja projektu



24 listopada 2024

Tymon Piwowarski

# Opis projektu

Projekt polegał na stworzeniu programu pozwalającego na wykonywanie operacji arytmetycznych oraz zamianę baz na nieograniczenie dużych liczbach naturalnych. Napisany został w języku C, wykorzystuje dynamiczną alokację pamięci.

Obsługiwane systemy liczbowe: **podstawy 2-16**

Dozwolone operacje arytmetyczne: **dodawanie, mnożenie, dzielenie, dzielenie modulo, potęgowanie**

# Sposób działania

Program jest przystosowany do działania w systemie Windows, uruchamiany z poziomu command prompt.

W celu uruchomienia programu należy wpisać komendę: **calc.exe <plik wejściowy> [plik wyjściowy]**

**Format pliku wejściowego**:

Tekstowy plik wejściowy zawiera listę działań arytmetycznych, które program ma wykonać. Działania arytmetyczne zapisane są w formacie: znak działania, spacja, podstawa systemu argumentów, lista argumentów oddzielonych pustymi liniami. W przypadku konwersji w linii działania zapisane są: podstawa systemu argumentów, spacja, podstawa systemu docelowego. Poszczególne działania rozdzielone są trzema liniami przerwy.

Znaki działań:

+ dodawanie,

\* mnożenie,

/ dzielenie całkowitoliczbowe,

% dzielenie modulo,

^ potęgowanie

**Format pliku wyjściowego**

W pliku wyjściowym zapisywane są podane na wejściu działania wraz z wstawionym wynikiem.

# Obsługa sytuacji wyjątkowych

1. W przypadku **niepodania nazwy pliku wejściowego** jako argumentu użytkownik zostaje o tym powiadomiony a program zwraca kod 1.
2. Kiedy w pliku wejściowym pojawi się **niedozwolony znak operacji bądź niedozwolony system liczbowy** zamiast wyników zostaną wypisane komunikaty „operacja niedozwolona”.
3. Kiedy w pliku wejściowym pojawi się **błędny argument tj. zawierający niedozwolone znaki lub cyfry spoza bieżącego systemu** zamiast wyników zostaną wypisane komunikaty „argument niepoprawny”.
4. Jeśli w pliku wejściowym podana zostanie **zbyt mała liczba argumentów** dla danej operacji (<1 dla zamiany systemów i < 2 dla arytmetyki) wypisane zostanie „argument niepoprawny”

Uwaga: program będzie starał się wykonać jak najwięcej operacji nawet, gdy część pliku wejściowego będzie zawierała niepoprawne dane.

# Implementacja

W programie do przechowywania liczb została wykorzystana struktura:  
„BigNum\_t” zawierająca:

- wektor liczb typu integer (jeden int to jedna cyfra)

- informację o liczbie cyfr

- informację o pojemności wektora

Liczby w programie reprezentowane są w systemie o podstawie 10e9, tak by jedna mieściła się w zmiennej typu int, a iloczyn dwóch w typie long long.

**Dodawanie i mnożenie:**

Algorytm dodawania i mnożenia zostały zaimplementowane jako zwykłe operacje wykonywane „pod kreską”

**Dzielenie i dzielenie modulo:**

Dzielenie bazuje na dzieleniu „pod kreską”. Algorytm działa w sposób następujący:

1. Przeskalowanie obydwu liczb tak, by pierwsza cyfra dzielnika była bliska 10^9
2. Dopisujemy 0 na początku liczby dzielonej.
3. Bierzemy n+1 pierwszych cyfr liczby dzielonej (gdzie n-liczba cyfr dzielnika)
4. Aproksymujemy wynik dzielenia poprzez uwzględnienie wyłącznie najbardziej znaczącej cyfry dzielnika.
5. Poprawiamy naszą aproksymacje poprzez powtarzane odejmowanie
6. Zapisujemy pierwszą cyfrę wyniku
7. Przenosimy resztę i dodajemy do niej kolejne cyfry liczby dzielonej
8. Powtarzamy kroki od 2. dopóki nie dojdziemy do ostatniej cyfry liczby dzielonej.

Reszta z dzielenia zostaje pozyskana przy wykorzystaniu algorytmu dzielenia całkowitego.

**Potęgowanie**

Potęgowanie bazuje na iteratywnej wersji algorytmu szybkiego potęgowania

**Zamiana systemów liczbowych**

Algorytm zamiany systemów (z systemu o podstawie A do B):

1. Zamień liczbę o podstawie A do systemu 10^9 (wykorzystując mnożenie i dodawanie kolejnych potęg liczby A)
2. Zamień liczbę o podstawie 10^9 do systemu B (wykorzystując powtarzanie operacji modulo B i zapisywanie wyników jako kolejnych cyfr)

# Podsumowanie

Projekt wymagał przemyślenia sposobu przechowywania dużych liczb, zdecydowałem się na interpretację ich jako liczby w bazie 10^9 ze względu na uproszczenie operacji mnożenia i dodawania (wyniki mieszczące się w long longach).

Jednym z problemów, które napotkałem to jak wolno wykonywane były operacje arytmetyczne, rozwiązałem to w sposób znalezienia sposobów na optymalizacje danych algorytmów i implementacje ich (np. szybkie potęgowanie, skalowanie liczb przy dzieleniu)