Dokumentacja kalkulatora dużych liczb

Jan Kaszycki

Listopad 2024

1 Podstawowe informacje dotyczące programu

1.1 Opis działania

Program wykonuje operacje arytmetyczne na nieujemnych liczbach całkowitych dowolnej wielkości w różnych systemach liczbowych ($2 \leqslant$ podstawa $\leqslant 16$) przy użyciu dynamicznej struktury danych - wektora. Obsługiwane operacje obejmują dodawanie, mnożenie, dzielenie całkowite, resztę z dzielenia oraz potęgowanie. Kod pozwala także na zmianę systemu liczbowego liczby.

1.2 Uruchomienie

Program powinien zostać uruchomiony z dwoma argumentami wywołania, czyli odpowiednio plikiem wejściowym i wyjściowym. Jeżeli nie podamy pliku wyjściowego, program utworzy plik wyjściowy dopisując out_ na początku nazwy pliku wejściowego. Jeżeli nie podamy ani pliku wejściowego ani wyjściowego, program będzie wczytywał ze standardowego wejścia i będzie wypisywał na standardowe wyjście.

1.3 Wejście

Na wejściu (czyli w pliku podanym jako argument lub w standardowym wejściu w przypadku braku pliku wejściowego) powinny znaleźć się operacje arytmetyczne do wykonania. Kolejne działania powinny być oddzielone trzema pustymi liniami. Tak powinno wyglądać działanie na wejściu:

Dla zmiany systemu:

```
<podstawa systemu liczby> <podstawa systemu docelowego>

<liczba>
```

Dla innych działań:

```
<znak działania> <podstawa systemu liczb>

<liczba 1>

<liczba 2>

<liczba 3>
...
```

1.4 Działania

Dostępne działania arytmetyczne powinny być opisywane odpowiednim znakiem:

dodawanie +

mnożenie $*$

dzielenie całkowite /

reszta z dzielenia %

potęgowanie ^

1.5 Wyjście

Na wyjście program zwróci wejście uzupełnione o wyniki odpowiednich działań (lub o odpowiednie kody błędów).

2 Struktury i funkcje

1. **Struktura danych - wektor**
   Wektor służy do przechowywania liczb. Składa się z pól:
   - data - wskaźnik na tablicę
   - size - ilość przechowywanych elementów
   - capacity - pojemność wektora
2. **Funkcje operujące na wektorze**
   - `vector_init(Vector *vec)`
     — inicjalizuje wektor.
   - `vector_free(Vector *vec)`
     — zwalnia pamięć zajmowaną przez wektor.

- `vector_resize(Vector *vec, size_t new_capacity)`
  — zmienia pojemność wektora.
- `vector_push_back(Vector *vec, int value)`
  — dodaje element na końcu wektora.
- `vector_size(Vector *vec)`
  — zwraca liczbę elementów w wektorze.
- `vector_reverse(Vector *vec)`
  — odwraca kolejność elementów w wektorze.
- `vector_copy(Vector *vec1, const Vector *vec2)`
  — kopiuje jeden wektor do drugiego (wraz z elementami, zawartością i rozmiarem)
- `ustawrozm(Vector *vec, size_t size)`
  — ustawia rozmiar wektora i wypełnia go zerami.
- `vector_copy_range`
  — kopiuje wycinek danych z jednego wektora do drugiego.

3. **Operacje arytmetyczne - funkcje**

- `dodaj`
  - Dodawanie pisemne liczb w dowolnym systemie
- `karacuba`
  - Mnożenie liczb za pomocą algorytmu Karacuby
- `podziel`
  - Dzielenie pisemne (poprzez odejmowanie z przesunięciem) liczb w dowolnym systemie
- `szybkie_potegowanie`
  - Potęgowanie szybkie wykorzystujące funkcję karacuba do mnożenia

4. **Zmiana systemu liczbowego - funkcja**

- `zmien_podstawe(Vector *vec1, int sys1, int sys2)`
  - Zmienia podstawę liczby z sys1 na sys2

5. **Funkcje pomocnicze**

- `odejmij`
  - Odejmowanie poprzez odwrócenie bitów liczby odejmowanej, dodanie liczb i odjęcie jedynki na odpowiedniej pozycji
- `usunzera`
  - Usuwa zera wiodące

- `rozszerz`
  - Zwiększa rozmiar liczby poprzez dopisanie zer wiodących
- `porownaj`
  - Porównuje dwie liczby o równej liczbie cyfr
- `porbezzer`
  - Porównuje dwie liczby nieposiadające zer wiodących
- `odejmij_na_poz`
  - Odejmowanie z przesunięciem (korzysta z funkcji odejmij i jest używane w funkcji podziel)
- `dzialanie`
  - Na podstawie wejścia decyduje, jakie funkcje wywołać

6. **Funkcje służące do wczytywania i wypisywania danych**
   - `czyt_liczbe`
     - Wczytuje liczbę z wejścia i zapisuje ją do wektora
   - `dopiszout`
     - Tworzy plik wyjściowy na podstawie pliku wejściowego
   - `wypisz`
     - Wypisuje liczbę zapisaną w wektorze na wyjście

3 Obsługa błędów

Błędy, po których wystąpieniu program kończy pracę :

- Nie udało się otworzyć pliku wejściowego
- Nie udało się otworzyć pliku wyjściowego

Błędy, po których program kontynuuje pracę od następnego działania :

- Dzielenie przez 0
- Niepoprawny znak działania
- Niepoprawna podstawa
- Liczba niezgodna z podstawą

Błędy w wejściu, pomimo których program wykona działanie :

- Zły odstęp między argumentami (ale mniejszy niż 3 puste linie)

4 Podsumowanie

Program można rozszerzyć o działania na liczbach ujemnych.
Program powinien działać zarówno w systemach Linux jak i na Windowsie

5 Wyjaśnienie: algorytm Karacuby

Algorytm Karacuby to działający rekurencyjnie algorytm
szybkiego mnożenia dużych liczb całkowitych
Jego złożoność wynosi $\Theta(n^{log_2(3)})$, przy mnożeniu dwóch n-cyfrowych liczb
Dokładne działanie oraz dowód złożoności można znaleźć w internecie np. na
stronie wikipedii