Projekt kalkulator

MacKon

rok 2023

Ogólne

Założenia projektu

- **Format pliku wejściowego:**
  Działania arytmetycznye zapisane w formie: znak działania, spacja, podstawa systemu argumentów, lista argumentów oddzielonych pustymi liniami oraz miejsca na wynik zaznaczone 3 pustymi liniami. W przypadku konwersji, w pierwszej linii działania zapisane są w postaci: podstawa systemu argumentów, spacja, podstawa systemu docelowego.

- **Format pliku wyjściowego:**
  W pliku wyjściowym zapisywany jest plik wejściowy z wynikami powstawianymi w miejsca na wynik zaznaczone 3 pustymi liniami.

- **Obsługiwane operacje:**

  1. dodawanie (+)
  2. mnożenie (*)
  3. dzielenie całkowitoliczbowe (/)
  4. dzielenie z resztą (%)
  5. potęgowanie (^)
  6. konwersja systemu liczbowego (oznaczana dwoma systemami liczbowymi oddzielonymi spacją)

- **Obsługiwane systemy liczbowe:**
  Dopuszczalne są systemy o podstawie od 2 do 16.

- **Uruchamianie programu:**

  – Program wczytuje dane z pliku wejściowego i zapisuje wyniki w pliku wyjściowym.
  – Nazwy plików podawane są jako argumenty wywołania.

- Jeśli nazwa pliku wyjściowego nie zostanie podana, program tworzy ją dostawiając na początku nazwy pliku wejściowego przedrostek "out_".
- Jeśli nazwa pliku wyjściowego nie istnieje, program tworzy plik o takiej nazwie.
- Zarówno plik wejściowy jak i wyjściowy zapisanę są w formacie TXT.
- Uruchomienie programu odbywa się poprzez wpisanie do terminala odpowiedniej komendy:
  "calc.exe ¡ścieżka do pliku wejściowego¿ [ścieżka do pliku wyjściowego]"

Implementacja programu:

Program napisany jest w języku C, skompilowany przez GCC i działa w systemie Windows.

Definicje

- **liczba** - liczba zapisana w postaci tablicy unsigned charów reprezentujących wartości poszczególnych cyfr, przekazywana w postaci wskaźnika na pierwszy element, rozmiaru tej liczby oraz rozmiaru buffora na tą liczbę, gdzie rozmiar buffora to miejsce zarezerwowane na liczbę. Dodanie cyfry do liczby polega na zwiększeniu rozmiaru i wpisaniu na ostatnie miejsce liczby odpowiadającej cyfrze jaką chcemy zapisać. W przypadku gdy po takiej operacji rozmiar buffora będzie mniejszy niż rozmiar, to rozmiar buffora zwiększa się dwukrotnie, a wskaźnik na pierwszy element reallocowany jest z ilością bajtów odpowiadającej nowej wartości rozmiaru buffora. Na początku rozmiar buffora równy jest 1.
  To, że argumentem funkcji jest liczba, oznacza, że do funkcji przekazywane są te 3 zmienne opisujące liczbę(w niektórych przypadkach nie wszystkie zmienne są przekazywane, gdyż nie jest to potrzebne). To, że liczba przekazywana jest do funkcji poprzez referencję, oznacza, że do funkcji przekazywane są te 3 zmienne poprzez referencję, oznacza to, że funkcja będzie mógła modyfikować rozmiar liczby w trakcie wykonywania. Zwykle liczby trzymane są odwrotnie.

- **miejsce na wynik** - miejsce w pliku przeznaczone do wpisania do niego wyniku, zwykle będące ciągiem 4 znaków nowej linii.

- **przykład** - część pliku wejsciowego zaczynająca się od operatora(bądź operatora zamiany systemów, czyli liczby) oraz systemu liczbowego, a kończąca dopiero miejscem na wynik tuż przed kolejmym przykładem, przykład może zawierać parę miejsc na wynik

- **część przykładu** - część przykładu składająca się z operatora, systemu liczbowego, kilku liczb oraz miejsca na wynik, przykładowo plik:

```
+ 10
```

```
1000

21

13



100

14
```

Będzie zawierał 1 przykład, ale 2 części przykładu, uwaga: obie części przykładu będą składały się z operatora: '+', systemu liczbowego: 10 i oczywiście odpowiadającym im liczb.

- **schemat** - tablica structów o nazwie blok składających się z chara i inta, gdzie char oznacza co znajduje sie w pliku w danym miejscu, a int oznacza dlugosc tego czegos. Możliwe chary w schemacie, to odpowiednio:
  - **l** - liczba
  - **s** - system liczbowy
  - **S** - system liczbowy będący jednocześnie operatorem
  - **z** - operator
  - **n** - znak nowej lini
  - **N** -miejsce na wynik
  - **m** - spacja, ale może być to też inny znak oddzielający, np. tabulator
  - **i** - inny znak do zignorowania, np. 'h'
  - **j** - znak do zigorowania, innt niż i, ostatecznie wszystkie n, m oraz i się na niego zamienią
  - **e** - koniec pliku

  Na przykład schemat pliku:

```
/ 4

0012312

00211
```

```
13 10

0120



123
```

Będzie wyglądał tak:

```
z       1
m       1
s       1
n       2
l       7
n       2
l       5
N       4
S       2
m       1
s       2
n       2
l       4
N       4
l       3
N       4
```

Opis rozwiązania

Procesowanie pliku przez program dzieli się na kilka części:

1. **Wczytywanie i przerabianie sciezek do pliku.** W tej części program wczytuje argumenty z cmd i wczytuje odpowiednie ścieżki do plików do odpowiednich tablic charów.

2. **Wczytywanie pliku wejsciowego do tablicy o nazwie plik.** W tej części program wczytuje zawartość pliku wejsciowego do tablicy o nazwie plik.

3. **Tworzenie schematu.** W tej części program bierze zawartość pliku wejsciowego i na jej podstawie tworzy schemat.

4. **Liczenie pliku.** Jest to największa część programu, w niej liczone są wyniki poszczególnych przykładów, całe liczenie pliku dzieje się w pętli powtarzającej się tyle razy ile jest przykładów. Sama pętla dzieli się na kilka części:

   4.1 **Wczytywanie znaku.** W tej części program wczytuje do chara najbliższy aktualnego miejsca operator. W przypadku, gdy operatorem jest zamiana systemów program wczytuje pierwszy system, a jako operator ustawia 'z'.

   4.2 **Wczytywanie systemu liczbowego.** W tej części program wczytuje do unsigned chara najbliższy system liczbowy.

   4.3 **Sprawdzanie ile jest 'N'.** W tej części program liczy ile w danym przykładzie jest miejsc na wypisanie wyniku.

   4.4 **Liczenie poszczególnych części przykladów.** W tej części porgram oblicza każdą część przykladu, robi to tyle razy ile jest 'N' w schemacie tego przykladu. Ta część też dzieli się na kilka kroków:

      4.4.1 **Wczytywanie pierwszego argumentu.** W tej części program wczytuje i odwraca pierwszą liczbę.

      4.4.2 **Obliczanie i wczytywanie kolejnych argumentów.** W tej część program wczytuje kolejne liczby, wykonuje dzialanie odpowiadające znakowi i wynik zapisuje w pierwszej liczbie. Ta część wykonywana jest tyle razy ile jest liczb poza pierwszą.

      4.4.3 **Wypisywanie** W tej części program wypisuje wynik, bądź też wiadomość o błędzie. Program wypisuje tablice charów od momentu, gdzie ostatnio skończył wypisywanie, a zamiast miejsca oznaczonego w schemacie 'N' wypisuje wynik. Wynik zapisany jest w pierwszej liczbie.

Funkcje

- **Ogólne**

  1. **swap** Funkcja typu void. Jej argumentami są 2 unsigned chary przekazywane poprzez referencję. Funkcja zamienia ze sobą wartości tych 2 argumentów.
  2. **zamienianie_kolejnosci** Funkcja typu void. Jej argumentami jest liczba. Funkcja zamienia kolejność cyfr tej liczby.
  3. **max** Funkcja typu int, pobiera 2 inty i zwraca większy z nich.
  4. **min** Funkcja typu int, pobiera 2 inty i zwraca mniejszy z nich.
  5. **usuwanie_zer** Funkcja typu void. Jej argumentem jest liczba przekazywana poprzez referencję. Funkcja usuwa zera z końca liczby (koniec liczby to zwykle początek, gdyż zwykle liczby trzymane są odwrotnie).
  6. **przesuniecie** Funkcja typu void. Jej argumentem jest liczba przekazana poprzez referencję. Funkcja usuwa pierwszy element liczby, a resztę elementów przesuwa o 1 na jego miejsce.

7. **charcpy** Funkcja typu void. Jej argumentem jest tablica charów. Funkcja kopiuje zawartość argumentu do globalnej tablicy charów służącej do obsługi błędów (więcej o tej tablicy będzie w zakładce obsługa błędów).

- **Arytmetyczne**

1. **dodawanie** Funkcja typu void. Jej argumentami są 2 liczby, przy czym pierwsza przekazywana jest poprzez referencję, obie liczby przekazywane są odwrócone, kolejnym argumentem jest system liczbowy. Funkcja dodaje 2 liczby w podanym systemie i wynik zapisuje odwrócony w 1 liczbie.
2. **mnożenie** Funkcja typu void. Jej argumentami są 2 liczby, przy czym pierwsza przekazywana jest poprzez referencję, obie liczby przekazywane są odwrócone, kolejnym argumentem jest system liczbowy. Funkcja mnoży 2 liczby w podanym systemie i wynik zapisuje odwrócony w 1 liczbie.
3. **odejmowanie** Funkcja typu void. Jej argumentami są 2 liczby, przy czym pierwsza przekazywana jest poprzez referencję, obie liczby przekazywane są odwrócone, kolejnym argumentem jest system liczbowy. Funkcja odejmuje 2 liczbę od 1, a wynik zapisuje odwrócony w 1 liczbie. Funkcja zakłada, że pierwsza liczba jest większa od drugiej.
4. **porow** Funkcja typu bool. Jej argumentami są 2 liczby, obie liczby przekazywane są odwrócone. Funkcja zwraca 0, gdy pierwszy argument jes $\leqslant$ od drugiego, a 1, gdy $>$.
5. **ile_razy_sie_miesci** Funkcja typu unsigned char. Jej argumentami są 2 liczby oraz systemliczbowy, obie liczby przekazywane są odwrócone. Funkcja zwraca ile razy drugi argument mieści się w pierwszym.
6. **reszta_z_ile_sie_miesci** Funkcja typu void. Jej argumentami są 2 liczby, obie liczby przekazywane są odwrócone, następnym argumentem jest system liczbowy oraz informacja ile razy drugi argument mieści się w pierwszym. Funkcja zapisuje w pierwszym argumencie resztę z dzielenia pierwszej liczby przez drugą w podanym systemie.
7. **reszta_z_dzielenia** Funkcja typu void. Jej argumentami są 2 liczby, przy czym pierwsza przekazywana jest poprzez referencję, obie liczby przekazywane są odwrócone, kolejnym argumentem jest system liczbowy. Funkcja zapisuje w pierwszym argumencie resztę z dzielenia pierwszego argumentu przez drugi w podanym systemie liczbowym.
8. **dzielenie** Funkcja typu void. Jej argumentami są 2 liczby, przy czym pierwsza przekazywana jest poprzez referencję, obie liczby przekazywane są odwrócone, kolejnym argumentem jest system liczbowy. Funkcja zapisuje w pierwszym argumencie wynik dzielenia całkowitoliczbowego w podanym systemie pierwszego argumentu przez drugi.
9. **mod2** Funkcja typu unsigned char. Jej argumentem jest liczba oraz system liczbowy. Funkcja zwraca resztę z dzielenia tej liczby przez 2 podanym systemie.

10. **dzielenie_przez_2** Funkcja typu void. Jej argumentem jest liczba przekazana poprzez referencję oraz system liczbowy. Funkcja w pierwszym argumencie zapisuje daną liczbę podzieloną przez 2.
11. **potegowanie** Funkcja typu void. Jej argumentami są 2 liczby przekazane poprzez referencję, obie liczby przekazywane są odwrócone, kolejnym argumentem jest system liczbowy. Funkcja w pierwszym argumencie zapisuje pierwszy argument do potęgi drugiego w podanym systemie liczbowym.
12. **zapisywanie_w_systemie** Funkcja typu void. Jej argumentem jest unsigned char oznaczający cyfrę, liczba oraz system liczbowy. Funkcja konwertuje cyfrę podaną w unsigned char do odpowiedniego systemu liczbowego i zapisuje ją w liczbie.
13. **zamienianie_systemow** Funkcja typu void. Jej argumentamem jest liczba przekazana odwrócona poprzez referencję oraz 2 systemy liczbowe. Funkcja zamienia liczbę z pierwszego systemu liczbowego do drugiego.

- **Wczytywanie**

1. **ascii_to_uchar** Funkcja typu unsigned char. Jej argumentem jest char. Funkcja zwraca wartość odpowiadającą wartości chara jako cyfry w systemie szesnastkowym. Przykładowo dla '3' zwraca 3, a dla 'B' zwraca 11. Dla małych liter funkcja zwraca takie same wartości co dla dużych.
2. **uchar_to_ascii** Funkcja typu char. Jej argumentem jest unsigned char. Funkcja zwraca literę odpowiadająca danej wartości unsigned chara w systemie szesnastkowym. Przykładowo dla 3 zwraca '3, a dla 11 zwraca 'B'.
3. **co_teraz** Funkcja typu char. Jej argumentem jest wskaźnik do tablicy charów oraz indeks. Funckja zwraca char odpowiadający znakowi będącemu indeks-1 znakiem w tablicy wejsciowej.
   Dla odpowiednich znaków funckja zwraca:
   - **l** - dla cyfry
   - **z** - dla operatora
   - **n** - dla znaku nowej lini
   - **m** - dla spacji, bądź też innego znaku oddzielającego, np. tabulatora
   - **i** - dla innego znakun np. 'h'
   - **e** - dla końca pliku
4. **przechodzenie_po_liczbie** Funkcja typu int. Jej argumentem jest wskaźnik do tablicy charów oraz indeks na pierwszą cyfrę liczby przekazany przez referencję. Funkcja zwraca długość liczby i zmienia wskaźnik na wskaźnik za ostatnią cyfrę.
5. **przechodzenie_po_enterze** Funkcja typu int. Jej argumentem jest wskaźnik do tablicy charów oraz indeks na pierwszy znak nowej lini przekazany przez referencję. Funkcja zwraca ilość znaków nowek linii i zmienia wskaźnik na wskaźnik za ostatnią mową linię.

6. **przechodzenie_po_spacji** Funkcja typu int. Jej argumentem jest wskaźnik do tablicy charów oraz indeks na pierwszy znak spacji przekazany przez referencję. Funkcja zwraca ilość znaków spacji i zmienia wskaźnik na wskaźnik za ostatnią spację.
7. **przechodzenie_po_i** Funkcja typu int. Jej argumentem jest wskaźnik do tablicy charów oraz indeks na pierwszą literę ciągu innych znaków przekazany przez referencję. Funkcja zwraca długość liczby i zmienia wskaźnik na wskaźnik ciąg innych znaków.
8. **szukanie_liczby** Funkcja typu unsigned char. Jej argumentem jest wskaźnik na pierwszy element schematu, indeks aktualnego położenia wskaźnika w schemacie przekazany poprzez referencję, rozmiar schematu, oraz bool czyjestznak przekazanmy przez referencję.
   - **gdy czyjesznak==0** funkcja zwraca 0 i zwiększa wskaźnik na najbliższą liczbę.
   - **gdy czyjestznak==1** funkcja zwraca różną wartość w zależności od tego na co natrafi pierwsze:
     * **liczba** funkcja zwraca 0 i zwiększa wskaźnik na pierwszą cyfrę znalezionej liczby.
     * **przerwę na wpisanie liczby, znak potem spację a potem liczbę** w takim wypadku funkcja zwraca 1, zwiększa wskaźnik na znak który znalazł oraz ustawia czyjestznak na 0. Funkcja ustawia wtedy też przerwę na 'N', znak na 'z', a liczbę na 's'.
     * **przerwę na wpisanie liczby, liczbę potem spację a potem liczbę** w takim wypadku funkcja zwraca 2, zwiększa wskaźnik na pierwszą liczbę którą znalazła oraz ustawia czyjestznak na 0. Funkcja ustawia wtedy też przerwę na 'N', pierwszą liczbę na 'S', a następną liczbę na 's'.
9. **wczytywanie_pliku_do_tablicy** Funkcja typu void. Jej argumentami są: wskaźnik na plik oraz tablica charów przekazana poprzez referencję jako wskaźnik na pierwszy element, rozmiar i rozmiar buffora. Funkcja wczytuje to co jest w pliku do tablicy.
10. **usuwanie** Funkcja typu void. Jej argumentami są: schemat przekazany poprzez referencję jako wskaźnik na pierwszy element, rozmiar i rozmiar buffora oraz indeks. Funkcja usuwa z schematu blok o indeksie podanym w argumencie i przesuwa pozostałe w lewo o 1.
11. **przerabianie_na_schemat** Funkcja typu void. Jej argumentami jest wskaźnik do tablicy charów z plikiem, oraz schemat przekazany przez referencję zapisany jako: wskaźnik na początek, rozmiar oraz rozmiar buffora. Funkcja bierze tablicę charów i zapisuje co w niej jest w schemacie.
    Sam proces zapisywania dzieli się na kilka części:

    11.1 **poczatkowe przerabianie** W tej części funkcja zamienia każdy operator na 'z', każdy ciąg cyfr na 'l', każdy ciąg znaków nowej linii na 'n', każdy

spacji na 'm', a każdy ciąg innych znaków na 'i', każdemu z tych ciągów funkcja przyporządkowywuje odpowiednią długość.

11.2 **zamienianie l na s, l na S, n na N, i na z w odpowiednich miejsach**
W tej części funkcja stara się przewidzieć, które: 'n' to tak naprawdę 'N', które: 'l' to tak naprawdę 's' lub 'S'.
Na początku funkcji deklarowane są 2 boole: czyjestznak oraz czyjestsystem oznaczające, czy w tym przykładzie już byl operator(bądź też 'S') i system liczbowy(operacja to zamiana systemów, to czy już był 2 system liczbowy).
Funkcja zamienia odpowiednie znaki za pomocą następujących reguł:

- **gdy wartość schematu teraźniejszego indeksu == 'l' i czyjestznak==1 i czyjestsystem==0**
  wtedy wartość schematu teraźniejszego indeksu = 's'
- **gdy wartość schematu teraźniejszego indeksu == 'l' i czyjestznak==1 i czyjestsystem==1**
  wtedy jeżeli następne bloki to: 'm' i 'l', to wartość schematu teraźniejszego indeksu = 'S'
- **gdy wartość schematu teraźniejszego indeksu == 'l' i czyjestznak==0**
  wtedy wartość schematu teraźniejszego indeksu = 'S'
- **gdy wartość schematu teraźniejszego indeksu == 'n' i dlugosc tej wartosci ¿=4 i czyjestznak==1 i czyjestsystem==1**
  wtedy wartość schematu teraźniejszego indeksu = 'N'
- **gdy wartość schematu teraźniejszego indeksu == 'n' i wskaźnik wskazuje na ostatni element schematu**
  wtedy wartość schematu teraźniejszego indeksu = 'N'
- **gdy wartość schematu teraźniejszego indeksu == 'n' i dlugosc tej wartosci ¿=4 i czyjestznak==1 i czyjestsystem==0**
  wtedy jeżeli następne bloki to 'z' lub kolejno 'l', 'm', 'l', wtedy wartość schematu teraźniejszego indeksu = 'N'
- **gdy wartość schematu teraźniejszego indeksu == 'n' i czyjestznak==1**
  wtedy jeżeli następne bloki to 'z' lub kolejno 'l', 'm', 'l', wtedy wartość schematu teraźniejszego indeksu = 'N'
- **gdy wartość schematu teraźniejszego indeksu == 'i' i czyjestznak==0 i poprzedni i następny blok to 'n' lub 'm'**
  wtedy wartość schematu teraźniejszego indeksu = 'z'
- **aktualny indeks to pierwszy blok schematu i następny blok to albo 'n' lub 'm'**
  wtedy wartość schematu teraźniejszego indeksu = 'z'

11.3 **n, m oraz i na j**
W tej części funkcja zamienia wszystkie 'n', 'm', 'i' na 'j'

11.4 **scalanie j ze sobą**
W tej części funkcja scala ze sobą wszystki 'j' - jak jest kilka 'j' obok siebie, to zamienia je na 1 'j' i odpowiednio zwiększa długość

11.5 **gdy ostatni element to nie N, to dodawanie N 0 na koncu**
W tej części gdy ostatni element schematu to nie 'N' to funkcja dodaje na koniec 'N' o długości 0.

Przykładowo dla przykładowego pliku:

```
/ 4

0012312

00211



13 10

0120



123
```

Po "poczatkowe przerabianie" schemat będzie wyglądał tak:

```
z       1
m       1
l       1
n       2
l       7
n       2
l       5
n       4
l       2
m       1
l       2
n       2
l       4
n       4
l       3
```

Po "zamienianie l na s, l na S, n na N, i na z w odpowiednich miejsach" będzie wyglądał tak:

```
z       1
m       1
s       1
n       2
l       7
n       2
l       5
N       4
S       2
m       1
s       2
n       2
l       4
N       4
l       3
```

Po "n, m oraz i na j" będzie wyglądał tak:

```
z       1
j       1
s       1
j       2
l       7
j       2
l       5
N       4
S       2
j       1
s       2
j       2
l       4
N       4
l       3
```

Po "scalanie j ze sobą" będzie wyglądał tak:

```
z       1
j       1
s       1
j       2
l       7
j       2
l       5
N       4
S       2
```

```
j       1
s       2
j       2
l       4
N       4
l       3
```

a po "gdy ostatni element to nie N, to dodawanie N 0 na koncu" będzie wyglądał tak:

```
z       1
j       1
s       1
j       2
l       7
j       2
l       5
N       4
S       2
j       1
s       2
j       2
l       4
N       4
l       3
N       0
```

12. **wczytywanie_znaku** Funkcja typu void. Jej argumentami są: wskaźnik na tablicę charów, rozmiar tej tablicy, indekst gdzie znajduje się znak i char przekazany przez referencję. Funkcja wpisuje znak znajdującym się pod danym indeksem do chara przekazanego przez referencję.
13. **wczytywanie_liczby** Funkcja typu void. Jej argumentami są: wskaźnik na tablicę charów, rozmiar tej tablicy, indekst gdzie znajduje się początek liczby oraz jej rozmiar i liczbę przekazaną przez referencję. Funkcja wpisuje liczbę zaczynającą się pod danym indeksem do liczby przekazanej przez referencję.
14. **wczytywanie_systemu_liczbowego** Funkcja typu bool. Jej argumentami są: wskaźnik na tablicę charów, rozmiar tej tablicy, indekst gdzie znajduje się początek systemu liczbowego oraz jego rozmiar i unsigned chara przekazanego przez referencję. Funkcja wpisuje system liczbowy zaczynający się pod danym indeksem do unsigned chara przekazanego przez referencję. Funkcja zwraca 0, gdy wczytywanie powiodło się i 1 gdy nie powiodło się

- **Wypisywanie**

1. **wypisywanie_liczby** Funkcja typu void. Jej argumentami są: wskaźnik na plik, liczba i ilość znaków nowej linii przeznaczonych na wpisanie w nie wyniku. Funkcja wpiuje liczbę z argumentu do aktualnego miejsca w pliku, robi to tak aby ilość znaków nowej linii była albo taka jak podana, albo 4.
2. **wypisywanie_bledu** Funkcja typu void. Jej argumentami są: wskaźnik na plik i ilość znaków nowej linii przeznaczonych na wpisanie w nie wyniku. Funkcja wpiuje wiadomość o błędzie z bledy[] (więcej o tej zmiennej będzie w zakładce: "Obsługa błędów") do aktualnego miejsca w pliku, robi to tak aby ilość znaków nowej linii była albo taka jak podana, albo 4.

Obsługa błędów

System obsługi błędów bazuje na globalnej tablicy i 2 zmiennych globalnych:

- **unsigned char** czyblad - zmienna przechowująca wiadomość o tym czy do tej pory w przykladzie wystapił błąd. Jej odpowiednie wartości oznaczają: 0 - brak błędu; 1 - błąd dotyczący tylko konkretnej części przykładu (części zakończonej polem na wpisanie wyniku); 2 - błąd dotyczący całego przykładu.
- **char** bledy[] - 100 elementowa tablica charów zawierająca wiadomość o błędzie
- **int** rb - rozmiar wiadomości o błędzie

Jeżeli czyblad==0 i wystąpi jakiś błąd to czyblad zostanie zmieniony, wiadomość o błędzie zostaje zapisana w bledy[], a w rb zostaje zapisana dlugosc wiadomosci.
Jeżeli podczas wypisywania czyblad!=0 to zamiast wyniku zostaje wypisana wiadomość o błędzie. Wprzypadku niektórych błędów od razu po błędzie zostaje wykonywana instrukcja **goto** do wypisywania.
Czyblad na pewno zeruje się po całym przykładzie. Gdy czyblad==1 na końcu części przykładu, wtedy czyblad również się zeruje. Ze względu na to wiadomości błędach odnoszących się do całego przykłądu(np. nieprawidłowy operator) zostaną wypisane we wszystkich miejscach na wpisanie, a wiadomości błędach odnoszących się tylko do części przykładu zostaną wypisane tylko do konkretnego miejsca na wpisanie. W przypadku wystąpienia kilku błędów wypisana zostanie tylko wiadomość o tym pierwszym.
Lista błędów opsługiwanych przez program:

1. **nieznany operator** blad występuje gdy użytkownik poda nieznany operator. Przykładowo dla wejscia:

   ```
   ? 10

   12

   13
   ```

Wynikiem będzie: "nieznany operator: '?'"

2. **nieznany system liczbowy** bład występuje gdy użytkownik poda system liczbowy nie mieszczący się w zakresie 2-16. Przykładowo dla wejscia:

   ```
   + 20

   7

   3
   ```

   Wynikiem będzie: "nieznany system liczbowy: 20"

3. **nieznana cyfra** bład występuje gdy użytkownik poda cyfrę inną niż cyfry z systemu szesnastkowego. Przykładowo dla wejscia:

   ```
   + 10

   7z3

   31
   ```

   Wynikiem będzie: "nieznana cyfra: 'z'"

4. **liczba nie miesci sie w systemie liczbowym** bład występuje gdy użytkownik poda cyfrę nie wykraczającą poza cyfry systemu szesnastkowego, ale za dużą dla systemu, który podał. Przykładowo dla wejscia: Wynikiem będzie: "liczba nie miesci sie w systemie liczbowym"

5. **brak danych** bład występuje gdy użytkownik nie poda odpowiedniej ilości danych do policzenia części przykładu. Przykładowo dla wejscia:

   ```
   /
   ```

   Wynikiem będzie: "brak danych"
   Oraz dla wejścia:

```
* 12
```

Wynikiem też będzie: "brak danych"

6. **błąd przy alokacji pamięci** blad występuje gdy podczas działania programu któryś malloc() lub realloc() zwróci NULLa, w takim przypadku:
   Wynikiem będzie: "błąd przy alokacji pamieci, być może liczba, którą podałeś jest za duża"

7. **brak sciezki do pliku** błąd występuje gdy użytkownik nie poda w terminalu żadnej ścieżki do pliku. Jako jedyna wiadomość o błędzie jest wypisywana w termianlu.

Program zinterpretuje także jakoś to co jest w pliku nawet jeżeli nie jest to końca takie jak wymaga tego specyfikacja. Przykładowo gdy przerwa na wynik jest większa niż 4 znaki nowej linii, to program i tak wpiszę tam wynik, a gdy jest za mała i widać że jest to miejsce na wynik, na przykład jest tuż przed kolejnym przykładem, to program zinterpretuje ją jako miejsce na wynik i powiększy ją do 4 znaków. Gdy natomiast system liczbowy nie będzie się znajdował po spacji po operatorze, to program zinterpretuje następną liczbę jako system liczbowy.