**1. Treść zadania**

Napisać program, który:

- odczyta operacje, system liczbowy i argumenty działania z pliku tekstowego „in.txt”

- wykona zadane obliczenia

- poda wynik obliczeń, w określonym przez użytkownika systemie liczbowym, zamieszczając go w pliku tekstowym o nazwie „out.txt” albo „out<numer przykładu.>.txt”.

**2. Specyfikacja zewnętrzna**

**2.1 Sposób uruchomienia programu**

Jednym ze sposobów na uruchomienie programu jest: zapisanie danych wejściowych w pliku tekstowym „in.txt”, który znajduje się w folderze Input włączenie pliku „Uruchom.exe”, który znajduje się w folderze Exe, następnie wybranie jednej z dwóch opcji zapisania odpowiedzi. Odpowiedzi zostaną zapisane w pliku tekstowym „out.txt” albo

„out<numer przykładu.>.txt”.Program pracuje w środowisku Visual Studio 2019. Aby go poprawnie uruchomić należy po włączeniu „projekt.c” użyć kombinacji klawiszowej „ctrl+5”.

**2.2 Kompilacja programu**

Program został skompilowany w Visual Studio 2019.

**3. Wprowadzone zmiany w programie**

Zmieniłem dzielenie w następujący sposób, aby przyśpieszyć, ponieważ poprzednim razem np. dla liczby 420-cyfrowej dzielonej przez cyfrę 210-cyfrową, czas w jakim zostałby podany wynik to około 10^200 dni, lecz wynik byłby poprawny. Obecnie lepszy algorytm pozwala nam na podanie wyniku w czasie niedłuższym niż kilka minut.

Wygląda on następująco: przykładowo dla liczby 123456789 dzielonej przez 3 w systemie 10-siętnym dopisujemy do 3 (9-1-1=)7 zer, w kolejnym kroku dodajemy do 0 wielokrotność 3\*10^7, dopóki będzie większa badź równa 123456789, gdy ten warunek nie będzie spełniony, czyli będzie większa albo równa dodajemy wielokrotność 3\*10^6, czyli usuwamy ostatnie 0. Powtarzamy te kroki do momentu, aż dotrzemy do 3 wówczas dodajemy wielokrotność 3, a następnie zapisujemy otrzymany wynik w zmiennej result oraz kończę działanie funkcji. Zostały również wprowadzone zmiany w raprocie, które zostały uzgodnione na wykładzie.

**4. Specyfikacja wewnętrzna**

**4.1 Struktura programu**

Struktury, stałe i zmienne zdefiniowane:

argumentp, argumentd – odpowiednio argument pierwszy oraz argument drugi dwie liczby naturalne

argumentx, argumenty – odpowiednio argument pierwszy oraz argument drugi po konwersji na dziesiątkowy system liczbowy

result – wynik operacji

syst – system liczbowy podany przez użytkownika

oper – operacja podana przez użytkownika, jaką należy wykonać

system – system liczbowy podany przez użytkownika, jeżeli konwersja ma być wynikiem działania

Funkcje:

void reverse(char\* x)

Funkcja zwraca element x w kolejności od ostatniego do pierwszego elementu.

is isbigger(char\* x, char\* y)

Funkcja zwraca jedynkę jeżeli element X jest większy od Y, dwójkę jeżeli Y jest większy od X albo zero jeżeli elementy są równe.

void change(char\* x, char\* y)

Funkcja zamienia element X z Y.

void add(char\* x, char\* y, char\* out)

Funkcja zwraca sumę X z Y, którą zapisuje, w zmiennej element char\* out.

void substract(char\* x, char\* y, char\* end)

Funkcja zwraca różnicę X z Y, którą zapisuje, w zmiennej element char\* end.

void multiple(char\* x, char\* y, char\* out)

Funkcja zwraca iloczyn X z Y, który zapisuje, w zmiennej char\* out.

void divide(char\* x, char\* y)

Funkcja zwraca iloraz X z Y, który zapisuje, w zmiennej globalnej wynik.

void modulo(char\* x, char\* y)

Funkcja zwraca X modulo Y, który zapisuje, w zmiennej globalnej wynik.

void power(char\* x, char\* y)

Funkcja zwraca wynik potęgowania (gdzie X jest podstawą, a Y wykładnikiem), który zapisuje w zmiennej globalnej wynik.

void conversion(int input, int output, char\* x, char\* out)

Funkcja zwraca wynik konwersji elementu X z systemu liczbowego input na system liczbowy output, wynik zapisuje w zmiennej char\* out.

int howlong(int x)

Funkcja zwraca długość liczby zamieniając wszystkie cyfry na 9.

**4.2 Użyte biblioteki**

Biblioteki poza standardowymi, które użyłem to: string.h.

**5. Opis operacji**

**"+"**

Dodawanie jest wykonywane w sposób pisemny.

**"^"**

Potęgowanie jest wykonane za pomocą mnożenia.

**"/"**

Dzielenie jest wykonane, w następujący sposób: przykładowo dla liczby 123456789 dzielonej przez 3 w systemie 10-siętnym dopisujemy do 3 (9-1-1=)7 zer, w kolejnym kroku dodajemy do 0 wielokrotność 3\*10^7, dopóki będzie większa badź równa 123456789, gdy ten warunek nie będzie spełniony, czyli będzie większa albo równa dodajemy wielokrotność 3\*10^6, czyli usuwamy ostatnie 0. Powtarzamy te kroki do momentu, aż dotrzemy do 3 wówczas dodajemy wielokrotność 3, a następnie zapisujemy otrzymany wynik w zmiennej result oraz kończę działanie funkcji.

**"%"**

Modulo jest wykonane w sposób identyczny, co dzielenie, do momentu aż dojdziemy do ostatniego kroku, wówczas odejmujemy od dzielnej maksymalną wielokrotność dzielnika.

**"\*"**

Mnożenie jest wykonane w sposób pisemny.

**6. Analiza Zadania**

**6.1 Wczytanie danych**

Zakładam, że plik tekstowy wraz z danymi do obliczenia, jest napisany tak, że w pierwszej linii znajduje się:

* operacja oraz system liczbowy, w jakim należy ją wykonać

lub

* system liczbowy danych wejściowych i system liczbowy, na który mamy wykonać konwersję argumentu

Kolejna linia jest linią odstępu, w trzeciej znajduje się pierwszy argument, czwarta również jest linią odstępu, a w piątej kolejny argument. Poprawność danych zostanie sprawdzona w późniejszym procesie programu. Z powodu, iż nie jest znana ilość operacji, jakie użytkownik chce wykonać, wszystkie dane są przechowywane w dynamicznie rezerwowanym obszarze pamięci operacyjnej. Każdy argument będzie zapisany na oddzielnie wyznaczonych miejscach w pamięci, aby móc się odwołać do każdej cyfry danego argumentu. Zakładam, że wszystkie dane zmieszczą się w pamięci operacyjnej.

**6.2 Obliczenia**

Na wstępie, po wczytaniu danych, sprawdzam jaka operacja została wprowadzona.

Jeżeli użytkownik chce otrzymać wynik operacji „\*”, „/”, „^”, „+”, „%”początkowo zostaje sprawdzona poprawność wpisanych argumentów, następnie jest wykonywana konwersja z systemu X na system 10., aby ograniczyć ilość funkcji w programie oraz doprowadzić do działań w systemie najbliższym człowiekowi. Po wykonaniu tej operacji wynik działania jest z powrotem konwertowany na system X, który został podany przez użytkownika.

W przeciwnym wypadku, jeżeli użytkownik chce skonwertować liczbę z systemu X na system Y, weryfikowana jest wstępnie poprawność argumentu, czy został dobrze wpisany. Następnie dokonywana jest konwersja.

**6.3 Zapis wyników**

Ostatnią częścią programu jest zapisanie podanych przez użytkownika przykładów wraz ich rozwiązaniami w pliku tekstowym o nazwie „out.txt”. Dane zostaną zapisane, podobnie, jak w pliku wejściowym, czyli w pierwszych pięciu linijkach zostaną podane dane wejściowe w szóstej linijka odstępu, a w siódmej wynik danego działania.

W przypadku podania błędnych danych wejściowych, np. znaki spoza systemu liczbowego, program wypisuje „Wprowadzono bledne dane dla przykladu: <numer przykładu> prosze poprawić”, natomiast gdy podane zostanie działanie niewykonalne, np. dzielenie przez zero, otrzymamy komunikat „Dzialanie niewykonalne: dzielenie przez zero”.

**7. Format danych wejściowych**

Plik z danymi wejściowymi powinien być plikiem tekstowym zapisanym w formacie ASCII, czyli jeden znak powinien zajmować jeden bajt. Dane należy podać jak w schemacie:

Jeśli chcemy wykonać działanie:

* znak działania

system liczbowy podanych argumentów

2 argumenty działania

Lub (jeśli chcemy dokonać konwersji)

* system liczbowy podanego argumentu

system, na który chcemy dokonać konwersji

argument, który ma zostać konwersji podany

Minimalnie wykorzystujemy jedną cyfrę, maksymalnie dziesięć cyfr oraz dwadzieścia cztery znaki wielkich liter. Przykładowe prawidłowe wprowadzenie danych:

\* 5

20234 43241230

15 7

ABACD43

Pierwsza operacja jest operacją mnożenia w systemie piątkowym argumentów 20234 oraz 43241230. Druga to konwersja z systemu piętnastkowego na siódemkowy liczby ABACD43.

**8. Obsługa programu**

Po uruchomieniu programu w górnej części pokaże się napis: „odpowiedź została zapisana w pliku odpowiedzi.txt”. Gdy dane w pliku tekstowym odpowiedzi.txt zostały źle wprowadzone, zamiast odpowiedzi zobaczymy napis „Wprowadzono błędne dane”, w przeciwnym wypadku otrzymamy n-odpowiedzi do naszych poprawnych n-operacji.

**9. Format danych wyjściowych**

Dane wyjściowe zapisują się w pliku tekstowym w formacie ASCII. Obliczone wartości będą są poprzedzone odpowiednimi przykładami z danych wejściowych, np.:

Weźmy pod uwagę przykład z podpunktu 3.2

Plik odpowiedzi.txt będzie wyglądał tak:

\* 5

20234 43241230

4331314420

15 7

ABACD43

50601030

**10. Uwagi i wnioski z testowania i uruchamiania**

a) Przetestowano program pod kątem ilości działań podanych przez użytkownika.

b) Program sprawdza zarówno poprawność wprowadzonych argumentów, jak i operacji oraz systemu liczbowego.

c) Nie przewidziano reakcji na sytuację, gdy stworzenie i zapisanie pliku wynikowego będzie niemożliwe np. ze względu na brak praw dostępu do zapisu w danym katalogu.

d) Nie przewidziano, że wartość zwracanej liczby przez jakąś funkcję w czasie tworzenia struktury dynamicznej przekroczy rozmiar dostępnej pamięci operacyjnej.

e) Nie przewidziano, że czas jest ograniczony, każda funkcja działa w czasie skończonym, wynik każdej operacji jest poprawny, lecz niektóre działają w długim czasie.

f) Nie przewidziano, że dane wejściowe, czyli argumenty pierwszy lub drugi, nie należą do zbioru liczb naturalnych.