# Projekt 2. Rafał Szczepanik 293471 1AR.

## Temat projektu:

Wykonać precyzyjny kalkulator, który może operować na liczbach do 70 cyfr. Działania ma realizować to: mnożenie, dodawanie, odejmowanie, silnia. Ma mieć możliwość korzystania z ostatniego wyniku. Wymagana sygnalizacja przekroczenia zakresu oraz komunikaty o błędach.

## Schemat blokowy programu:

START

TAK

Zakończ?

KONIEC

NIE

Wyświetlenie wyniku

Wprowadzenie znaku działania jakie ma zostać wykonane

czy pierwsza liczba   
jest ujemna?

Wprowadzenie znaku i wartości drugiej liczby z możliwością jej korekty lub użycia ostatniego wyniku

Wprowadzenie znaku i wartości pierwszej liczby z możliwością jej korekty lub użycia ostatniego wyniku

Wykonanie działania\*

NIE

TAK

NIE

czy znak to „!”?

TAK

Ustalenie jakie działanie ma zostać wykonane oraz znak wyniku na podstawie znaków pierwszej i drugiej liczby oraz znaku działania\*

Komunikat o błędzie

Wykonanie działania\*

Wynik zbyt długi?

Komunikat o błędzie

NIE

\*Operacje opisane   
w kolejnych punktach.

## Zmienne globalne i makrodefinicje :

|  |  |
| --- | --- |
| #define MAX 70 | zakres długości liczb. |
| struct number{  char value[MAX+2];  char sign;  char end; } | struktura – liczba tablica znakowa przechowująca wartość liczby zmienna znakowa przechowująca znak liczby (+ lub -) zmienna pomocnicza przy wczytywaniu liczby, w przypadku wyniku flaga zbyt dużej ilości cyfr; |
| number result; | zmienna przechowująca wynik, na niej wykonywane są operacje przez funkcje wykonywania działań |

## Opis wybranych operacji ze schematu blokowego programu:

1. Wykonanie działania odbywa się za pomocą 4 funkcji:
   1. void add(char fnumb[],char snumb[]) – dodawanie dwóch liczb
   2. void sub(char fnumb[],char snumb[]) – odejmowanie pierwszej liczby od drugiej (pierwsza liczba musi być większa od drugiej)
   3. void mul(char fnumb[],char snumb[]) – mnożenie dwóch liczb
   4. void fac(char fnumb[]) - silnia liczby

Wszystkie powyższe funkcje zbudowane są analogicznie. Jako parametry przyjmują tablice znakowe (jedną lub dwie) wypełnione cyframi od 1 do 9. W przypadku funkcji sub pierwsza liczba musi być większa od drugiej. Sprawdzenie poprawności danych odbywa się na poziomie implementacji funkcji. Funkcje powodują zmianę globalnej zmiennej – char result.value[MAX+2], która wyświetlana jest jako wynik działania w końcowej fazie działania programu.

**Schemat blokowy funkcji: void add(char fnumb[], char snumb[]):**

START

długość wyniku > MAX

TAK

Zmień flagę błędu wyniku.  
Wynik = 0;

NIE

KONIEC

Usuń zera z końca wyniku.

Odwróć kolejność cyfr w pierwszej   
i drugiej liczbie oraz w wyniku.

NIE

TAK

dodaj cyfry z i-tego miejsca dwóch liczb i cyfrę z i-tego miejsca wyniku wpisz cyfrę jedności na i-te miejsce wyniku oraz cyfrę dziesiątek na i+1 miejsce wyniku.

i++.

i<=długości 1. liczby lub  
 i<=długość 2. liczby

Odwrócenie kolejności cyfr w pierwszej  
i drugiej liczbie.

Ustawienie wartości wyniku na 0.

i = 0;

1. Ustalenie jakie działanie ma zostać wykonane oraz ustalenie znaku wyniku dokonuje się przy pomocy instrukcji warunkowych case i if. Schemat blokowy tego fragmentu kodu:

sym – znak działania; zn. 1. l. – znak 1. liczby; zn. wyn. – znak wyniku.

START

sym== „-”  
i zn 2. l. == „-”

NIE

TAK

sym== „+”  
i zn 2. l. == „-”

TAK

sym= „-”; zn. 2. l. = „+”

sym==„+”; zn. 2. l. = „+”

NIE

NIE

TAK

sym== „+”

NIE

sym== „-”

TAK

zn. 1. l. == „+”

TAK

TAK

zn. 1. l. == zn. 2. l.

NIE

zn. 1. l. == „+”

Odejmij dwie liczby;  
Zn. wyn. w zależności od tego, która większa.

NIE

Dodaj dwie liczby;  
Zn. wyn. = „+”

TAK

Dodaj dwie liczby;  
Zn. wyn. = „-”

Odejmij dwie liczby;  
Zn. wyn. w zależności od tego, która większa.

Pomnóż 2 liczby;

Zn. wyn. = „-”

Pomnóż 2 liczby;

Zn. wyn. = „+”

KONIEC

## Pozostałe funkcje użyte w programie:

|  |  |
| --- | --- |
| **void reverse(char fnumb[])** | jako parametr przyjmuje tablicę znakową dowolnej długości  i odwraca kolejność znaków w tej tablicy. |
| **void copy(char fnumb[], char snumb[])** | jako parametr przyjmuje dwie tablice znakowe dowolnej długości i kopiuje zawartość pierwszej tablicy do drugiej. |
| **void clearTab(char fnumb[], int size)** | jako parametr przyjmuje tablicę znakową oraz jej rozmiar  i na każdym miejscu jako wartość wpisuje 0. |
| **void clearTab(number fnumb, number snumb, char symb)** | jako parametry przyjmuje dwie struktury number oraz znak i wypisuje na konsolę zgodnie z algorytmem: znak pierwszej liczby, pierwsza liczba, znak, znak drugiej liczby, druga liczba. Konieczna z uwagi na błędy związane z wypisywaniem znaku ”0” w bibliotece ncurses. |

## Biblioteki użyte w programie:

Oprócz standardowych bibliotek (<stdio.h>,<stdlib.h>) w programie użyto bibliotek: <string.h>, w celu użycia funkcji: strlen(), strcmp()) oraz biblioteki <ncurses.h>, w celu użycia trybu wprowadzania „bez entera” oraz podświetlania błędów na czerwono i wyniku na zielono.