Kalkulator wyrażeń zapisanych w ONP

AUTHOR
Wersja
Cz, 27 maj 2021

Spis treści

Table of contents

Strona główna

Kalkulator Odwrotnej Notacji Polskiej

Program obliczający wyrażenia zapisane w odrwotnej notacji polskiej.

Autor

Stanisław Królikiewicz

Data

2021.05.26

Wersja

1.0

Kontakt:

stanislaw.krolikiewicz.stud@pw.edu.pl

Odwrotna notacja polska w skrócie ONP jest algorytmem stosowanym do zapisu wyrażeń arytmetycznych bez użycia nawiasów. Powstał on na podstawie notacji polskiej stworzonej przez polskiego matematyka Jana Łukasiewicza.

Indeks klas

Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Indeks plików

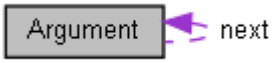
Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Dokumentacja klas

Dokumentacja struktury Argument

Element listy dynamicznej.
`#include <lista.h>`
Diagram współpracy dla Argument:



Atrybuty publiczne

- double **x**
  Wartość elementu.

- **Argument * next**

Opis szczegółowy

Element listy dynamicznej.

Definicja elementu jednokierunkowej listy argumentów, która docelowo ma służyć ajko stos.

Dokumentacja atrybutów składowych

Argument* Argument::next

double Argument::x

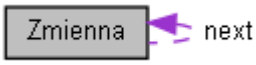
Wartość elementu.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- **lista.h**

Dokumentacja struktury Zmienna

Element listy dynamicznej.
`#include <lista.h>`
Diagram współpracy dla Zmienna:



Atrybuty publiczne

- string **nazwa**
  Nazwa zmiennej.

- double **wartosc**
  Wartość liczbowa przechowywana przez zmienną

- **Zmienna * next**
  Wskaźnik na następny element listy.

Opis szczegółowy

Element listy dynamicznej.

Definicja elementu jednokierunkowej listy zmiennych.

Dokumentacja atrybutów składowych

string Zmienna::nazwa

Nazwa zmiennej.

Zmienna* Zmienna::next

Wskaźnik na następny element listy.

double Zmienna::wartosc

Wartość liczbowa przechowywana przez zmienną

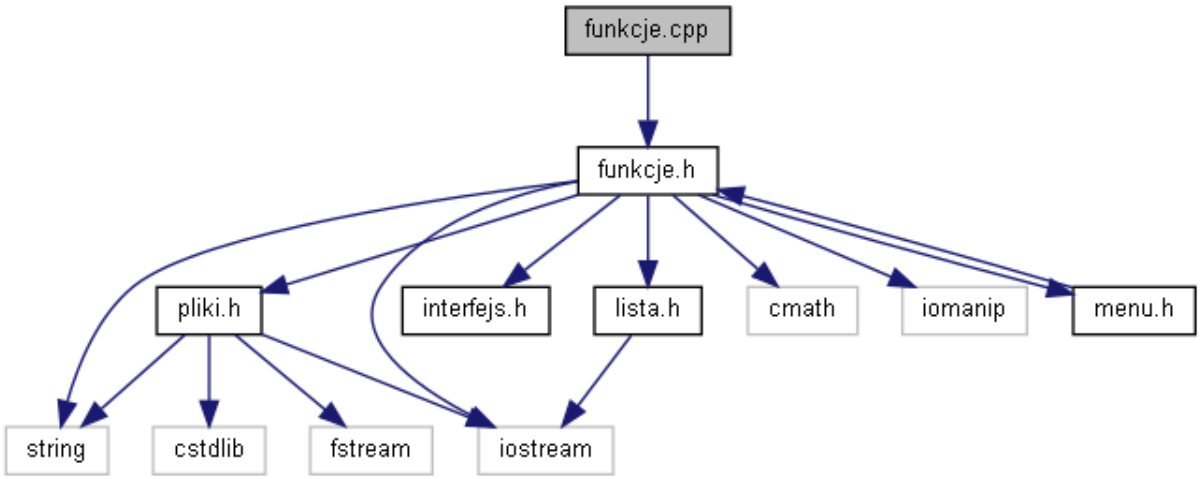
Dokumentacja dla tej struktury została wygenerowana z pliku:

- **lista.h**

Dokumentacja plików

Dokumentacja pliku funkcje.cpp

Plik implementacji modułu Funkcje.
`#include "funkcje.h"`
Wykres zależności załączania dla funkcje.cpp:



Funkcje

- void **modyfikujWyrazenie** ()
  Funkcja modyfikująca wyrazenie.

- void **obliczWyrazenie** ()
  Funkcja obliczająca wyrazenie.

- void **modyfikujZmienne** ()
  Funkcja modyfikująca zmienne.

- double **zamienStringNaDouble** (string napis)
  Zamień napis na liczbę

- void **odlozNaStos** (double x)
  Funkcja odkłada liczbę na stos.

- double **jakiOperator** (char a, bool &zero)
  Funkcja, która interpetuje operator.

- void **czyToZmienna** (bool &f, bool &t, double &val, char T[], int n, int &i, double &wynik)
  Funkcja sprawdza, czy zawarty w wyrażeniu ciąg znaków jest zmienną

- void **czySinCos** (bool &f, string &name, char T[], int n, int i)
  Funkcja sprawdza, czy zawarty w wyrazeniu ciąg znaków jest operatorem sinus lub cosinus.

- double **obliczSinCos** (string name)
  Funkcja obliczająca funkcje trygonometryczną

- void **dodajZmienna** (string name, double value, bool &c)
  Funkcja dodająca zmienne.

- void **edytujZmienna** (int j)
  Funkcja edytująca zmienne.

Zmienne

- const string **nazwa_pliku** = "wyrazenie.txt"
  ustalanie sciezki do pliku przechowującego wyrazenie

- **Argument * stos** = nullptr
  inicjowanie stosu

- **Zmienna** * **listaZmiennych** = nullptr
  inicjowanie listy dynamicznej jednokierunkowej zmiennych

Opis szczegółowy

Plik implementacji modułu Funkcje.

Zawiera on definicje najbardziej istotnych funkcji programu, służących do obliczania i edytowania wyrażenia oraz tworzenia i modyfikacji zmiennych.

Dokumentacja funkcji

void czySinCos (bool & *f*, string & *name*, char *T*[], int *n*, int *i*)

Funkcja sprawdza, czy zawarty w wyrazeniu ciąg znaków jest operatorem sinus lub cosinus.

Definicja funkcji, która określa, czy dany ciąg znaków jest operatorem sinus lub cosinus.

Parametry

&f	flaga, która informuje, czy badany ciąg znaków w wyrażeniu jest sinusem lub cosinusem
&name	zmienna, która przechowuje wartość znalezionego operatora (sin lub cos)
T[]	talica, która przechowuje kolejne znaki wyrazenia
n	rozmiar tablicy
i	nr pierwszego wyrazu ciagu

void czyToZmienna (bool & *f*, bool & *t*, double & *val*, char *T*[], int *n*, int & *i*, double & *wynik*)

Funkcja sprawdza, czy zawarty w wyrażeniu ciąg znaków jest zmienną

Definicja funkcji, która sprawdzaa czy zawarte w wyrażeniu ciągi wyrazów są zmiennymi wprowadzonymi wcześniej przez użytkownika.

Parametry

&f	flaga, która daje informacje, czy istnieje zmienna o podanej nazwie
&t	
&val	
T[]	tablica znakow, które połączeniu tworzą szukaną nazwę
n	rozmiar tablicy
&i	nr pierwszego znaku ciągu wyrazów w tablicy
&wynik	zapisuje wynik, w przypadku, gdy ciąg jest sinusem lub cosinusem

void dodajZmienna (string *name*, double *value*, bool & *c*)

Funkcja dodająca zmienne.

Definicja funkcji, która umożliwia dodawanie nowych zmiennych.

Parametry

name	nazwa nowej zmienej
value	wartosc liczbowa nowej zmiennej
&c	flaga, która informuje, czy zmienna o podanej nazwie już istnieje

void edytujZmienna (int *j*)

Funkcja edytująca zmienne.

Definicja funkcji, która edytuje wybraną zmienną.

Parametry

i	nr zmiennej

double jakiOperator (char *a*, bool & *zero*)

Funkcja, która interpetuje operator.

Definicja funkcji, która interpretuje jaki opersator matematyczny spośród '+', '-', '/', '*' lub '^' ma być użyty.

Parametry

a	znak, który ma zostać zinterpretowany
&zero	flaga, która ma wskazywać, czy operator dzielenie ma dzielić przez zero

Zwraca

liczba zwracana jako wartosc wyniku dzialania z uzyciem operatora

void modyfikujWyrazenie ()

Funkcja modyfikująca wyrazenie.

Definicja funkcji, która daje możliwość modyfikacji wyrazenia zapisanego w pliku tekstowym, które następnie jest obliczane.

void modyfikujZmienne ()

Funkcja modyfikująca zmienne.

Definicja funkcji, daje możliwość modyfikacji lub/i dodawania zmienncyh przez użytkownika.

double obliczSinCos (string *name*)

Funkcja obliczająca funkcje trygonometryczną

Definicja funkcji, która oblicza sinus lub cosinus wartości.

Parametry

name	nazwa funkcji trygonometrycznej

void obliczWyrazenie ()

Funkcja obliczająca wyrazenie.

Definicja funkcji, która oblicza wyrażenie zapisane w pliku tekstowym.

void odlozNaStos (double *x*)

Funkcja odkłada liczbę na stos.

Definicja funkcji, która odkłada podaną wartość (double) na stos, tak aby w późniejszym etapie wartość została pobrana zmanipulowana przed następujący po niej operator.

Parametry

x	liczba odkladana na stos jako wartos nowego elementu listy argumentów

double zamienStringNaDouble (string *napis*)

Zamień napis na liczbę

Definicja funkcji, która zamienia napis przedstawiający jakąś liczbę na zmienną typu double.

Parametry

napis	reprezentuje ciąg cyfr, ale jeszcze nie jest liczbą

Zwraca

x liczba, która jest już typu double

Dokumentacja zmiennych

Zmienna* listaZmiennych = nullptr

inicjowanie listy dynamicznej jednokierunkowej zmiennych

const string nazwa_pliku = "wyrazenie.txt"

ustalanie sciezki do pliku przechowującego wyrazenie

Argument* stos = nullptr

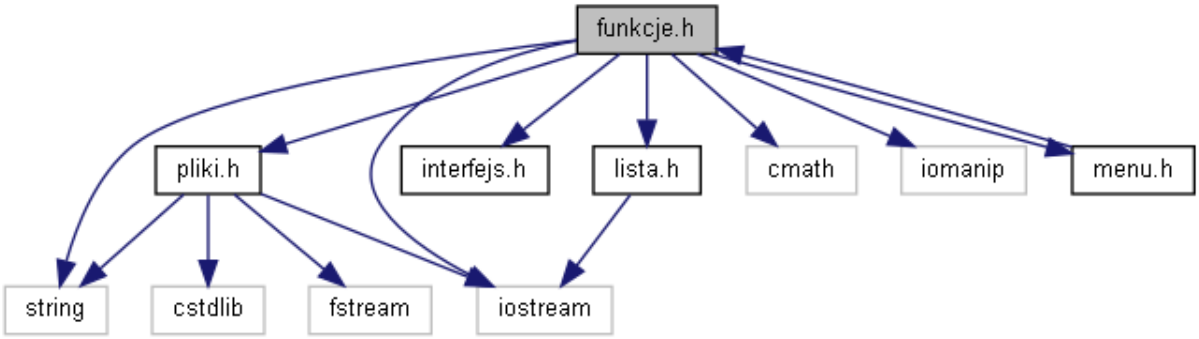
inicjowanie stosu

Dokumentacja pliku funkcje.h

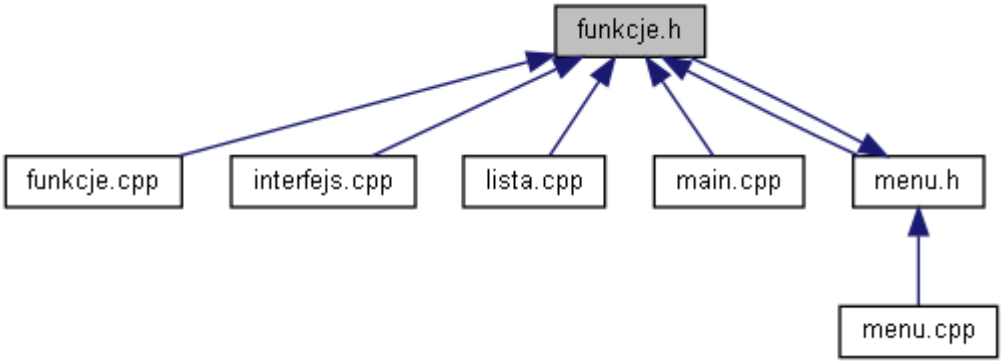
Plik nagłówkowy modułu Funkcje.

```
#include "pliki.h"
#include "interfejs.h"
#include "lista.h"
#include <iostream>
#include <string>
#include <cmath>
#include <iomanip>
#include "menu.h"
```

Wykres zależności załączania dla funkcje.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- void **modyfikujWyrazenie** ()
  Funkcja modyfikująca wyrazenie.

- void **obliczWyrazenie** ()
  Funkcja obliczająca wyrazenie.

- void **modyfikujZmienne** ()
  Funkcja modyfikująca zmienne.

- double **zamienStringNaDouble** (string napis)
  Zamień napis na liczbę

- void **odlozNaStos** (double x)
  Funkcja odkłada liczbę na stos.

- double **jakiOperator** (char a, bool &zero)
  Funkcja, która interpetuje operator.

- void **czyToZmienna** (bool &f, bool &t, double &val, char T[], int n, int &i, double &wynik)
  Funkcja sprawdza, czy zawarty w wyrażeniu ciąg znaków jest zmienną

- void **czySinCos** (bool &f, string &name, char T[], int n, int i)
  Funkcja sprawdza, czy zawarty w wyrazeniu ciąg znaków jest operatorem sinus lub cosinus.

- double **obliczSinCos** (string name)
  Funkcja obliczająca funkcje trygonometryczną

- void **dodajZmienna** (string name, double value, bool &c)
  Funkcja dodająca zmienne.

- void **edytujZmienna** (int i)
  Funkcja edytująca zmienne.

Opis szczegółowy

Plik nagłówkowy modułu Funkcje.

Zawiera on deklaracje najbardziej istotnych funkcji programu, służących do obliczania i edytowania wyrażenia oraz tworzenia i modyfikacji zmiennych.

Dokumentacja funkcji

void czySinCos (bool & *f*, string & *name*, char *T*[], int *n*, int *i*)

Funkcja sprawdza, czy zawarty w wyrazeniu ciąg znaków jest operatorem sinus lub cosinus.

Deklaracja funkcji, która ma na celu określenie, czy dany ciąg znaków jest operatorem sinus lub cosinus.

Parametry

&f	flaga, która informuje, czy badany ciąg znaków w wyrażeniu jest sinusem lub cosinusem
&name	zmienna, która przechowuje wartość znalezionego operatora (sin lub cos)
T[]	talica, która przechowuje kolejne znaki wyrazenia
n	rozmiar tablicy
i	nr pierwszego wyrazu ciagu

Definicja funkcji, która określa, czy dany ciąg znaków jest operatorem sinus lub cosinus.

Parametry

&f	flaga, która informuje, czy badany ciąg znaków w wyrażeniu jest sinusem lub cosinusem
&name	zmienna, która przechowuje wartość znalezionego operatora (sin lub cos)

T[]	talica, która przechowuje kolejne znaki wyrazenia
n	rozmiar tablicy
i	nr pierwszego wyrazu ciagu

void czyToZmienna (bool & *f*, bool & *t*, double & *val*, char *T*[], int *n*, int & *i*, double & *wynik*)

Funkcja sprawdza, czy zawarty w wyrażeniu ciąg znaków jest zmienną

Deklaracja funkcji, która ma na celu, sprawdzić, czy zawarte w wyrażeniu ciągi wyrazów są zmiennymi wprowadzonymi wcześniej przez użytkownika.

Parametry

&f	flaga, która daje informacje, czy istnieje zmienna o podanej nazwie
&t	
&val	
T[]	tablica znakow, które połączeniu tworzą szukaną nazwę
n	rozmiar tablicy
&i	nr pierwszego znaku ciągu wyrazów w tablicy
&wynik	zapisuje wynik, w przypadku, gdy ciąg jest sinusem lub cosinusem

Definicja funkcji, która sprawdzaa czy zawarte w wyrażeniu ciągi wyrazów są zmiennymi wprowadzonymi wcześniej przez użytkownika.

Parametry

&f	flaga, która daje informacje, czy istnieje zmienna o podanej nazwie
&t	
&val	
T[]	tablica znakow, które połączeniu tworzą szukaną nazwę
n	rozmiar tablicy
&i	nr pierwszego znaku ciągu wyrazów w tablicy
&wynik	zapisuje wynik, w przypadku, gdy ciąg jest sinusem lub cosinusem

void dodajZmienna (string *name*, double *value*, bool & *c*)

Funkcja dodająca zmienne.

Deklaracja funkcji, która ma umożliwiać dodawanie nowych zmiennych.

Parametry

name	nazwa nowej zmienej
value	wartosc liczbowa nowej zmiennej
&c	flaga, która informuje, czy zmienna o podanej nazwie już istnieje

Definicja funkcji, która umożliwia dodawanie nowych zmiennych.

Parametry

name	nazwa nowej zmienej
value	wartosc liczbowa nowej zmiennej
&c	flaga, która informuje, czy zmienna o podanej nazwie już istnieje

void edytujZmienna (int *j*)

Funkcja edytująca zmienne.

Deklaracja funkcji, która edytuje wybraną zmienną.

Parametry

i	nr zmiennej

Definicja funkcji, która edytuje wybraną zmienną.

Parametry

i	nr zmiennej

double jakiOperator (char *a*, bool & *zero*)

Funkcja, która interpetuje operator.

Deklaracja funkcji, która ma interpretować jaki operator matematyczny spośród '+', '-', '/', '*' lub '^' ma być użyty.

Parametry

a	znak, który ma zostać zinterpretowany
&zero	flaga, która ma wskazywać, czy operator dzielenie ma dzielić przez zero

Zwraca

liczba zwracana jako wartosc wyniku dzialania z uzyciem operatora

Definicja funkcji, która interpretuje jaki opersator matematyczny spośród '+', '-', '/', '*' lub '^' ma być użyty.

Parametry

a	znak, który ma zostać zinterpretowany
&zero	flaga, która ma wskazywać, czy operator dzielenie ma dzielić przez zero

Zwraca

liczba zwracana jako wartosc wyniku dzialania z uzyciem operatora

void modyfikujWyrazenie ()

Funkcja modyfikująca wyrazenie.

Deklaracja funkcji, która daje możliwość modyfikacji wyrazenia zapisanego w pliku tekstowym, które następnie jest obliczane.

Definicja funkcji, która daje możliwość modyfikacji wyrazenia zapisanego w pliku tekstowym, które następnie jest obliczane.

void modyfikujZmienne ()

Funkcja modyfikująca zmienne.

Deklaracja funkcji, która ma dawać możliwość modyfikacji lub/i dodawania zmienncyh przez użytkownika.

Definicja funkcji, daje możliwość modyfikacji lub/i dodawania zmienncyh przez użytkownika.

double obliczSinCos (string *name*)

Funkcja obliczająca funkcje trygonometryczną

Deklaracja funkcji, która oblicza sinus lub cosinus wartości.

Parametry

name	nazwa funkcji trygonometrycznej

Definicja funkcji, która oblicza sinus lub cosinus wartości.

Parametry

name	nazwa funkcji trygonometrycznej

void obliczWyrazenie ()

Funkcja obliczająca wyrazenie.

Deklaracja funkcji, która ma obliczać wyrażenie zapisane w pliku tekstowym.

Definicja funkcji, która oblicza wyrażenie zapisane w pliku tekstowym.

void odlozNaStos (double *x*)

Funkcja odkłada liczbę na stos.

Deklaracja funkcji, która ma odkładać podaną wartość (double) na stos, tak aby w późniejszym etapie wartość została pobrana zmanipulowana przed następujący po niej operator.

Parametry

x	liczba odkladana na stos jako wartos nowego elementu listy argumentów

Definicja funkcji, która odkłada podaną wartość (double) na stos, tak aby w późniejszym etapie wartość została pobrana zmanipulowana przed następujący po niej operator.

Parametry

x	liczba odkladana na stos jako wartos nowego elementu listy argumentów

double zamienStringNaDouble (string *napis*)

Zamień napis na liczbę

Deklaracja funkcji, która ma zmieniać napis przedstawiający jakąś liczbę na zmienną typu double.

Parametry

napis	reprezentuje ciąg cyfr, ale jeszcze nie jest liczbą

Zwraca

x liczba, która jest już typu double

Definicja funkcji, która zamienia napis przedstawiający jakąś liczbę na zmienną typu double.

Parametry

napis	reprezentuje ciąg cyfr, ale jeszcze nie jest liczbą

Zwraca

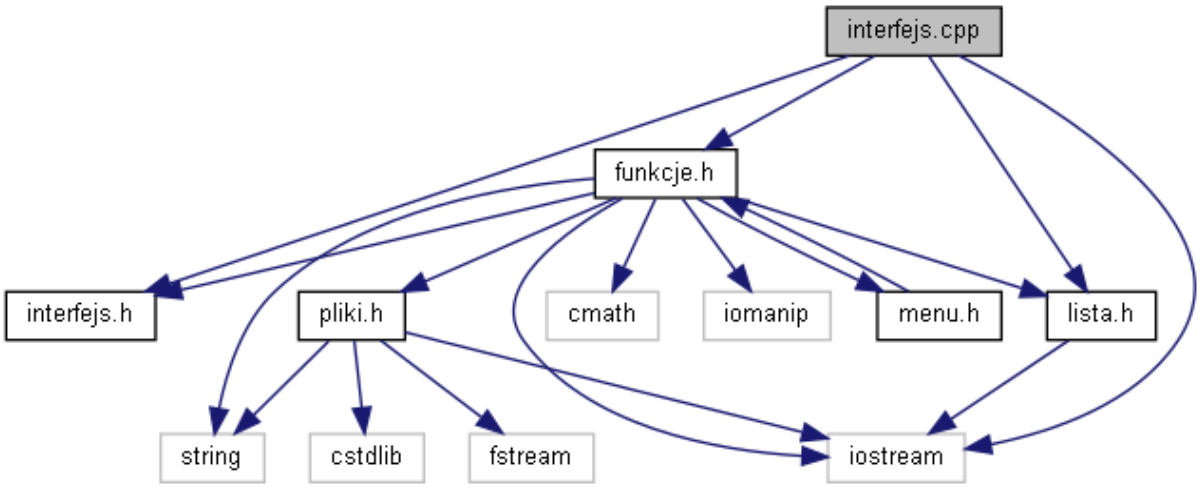
x liczba, która jest już typu double

Dokumentacja pliku interfejs.cpp

Plik implementacji modułu Interfejs.

```
#include "interfejs.h"
#include "funkcje.h"
#include "lista.h"
#include <iostream>
```

Wykres zależności załączania dla interfejs.cpp:



Funkcje

- void **intro** ()
  Intro.

- void **wrocDoMenu** ()
  Powrót do Menu.

- void **wrocDoModyfikacji** ()
  Powrót do wyboru opcji modyfikacji zmiennych.

- void **exit** ()
  Exit.

Opis szczegółowy

Plik implementacji modułu Interfejs.

Zawiera on definicje podstawowych funkcji interfejsu, zawartych w pliku nagłowkowym, które mają na celu usprawnienie poruszania się użytkownika po aplikacji

---

Dokumentacja funkcji

void exit ()

Exit.

Deklaracja funkcji, która kończy działanie programu.

Funkcja, która kończy działanie programu.

Czyszczenie ekranu.

Zmienna a , która przechowuje znak wpisany przez użytkownika.

Pętla, która ma powtarza się dopóki użytkownik nie zdeycduje się, czy chce zakończyc działanie programu, czy nie.

void intro ()

Intro.

Deklaracja funkcji intro.

Funkcja, która wita użytkownika w Kalkulatorze ONP.

void wrocDoMenu ()

Powrót do Menu.

Deklaracja funkcji, która wraca do menu głównego.

Funkcja, która cofa użytkownika do menu głównego.

void wrocDoModyfikacji ()

Powrót do wyboru opcji modyfikacji zmiennych.

Deklaracja funkcji, która wraca do wyboru opcji modyfikowania zmiennych.

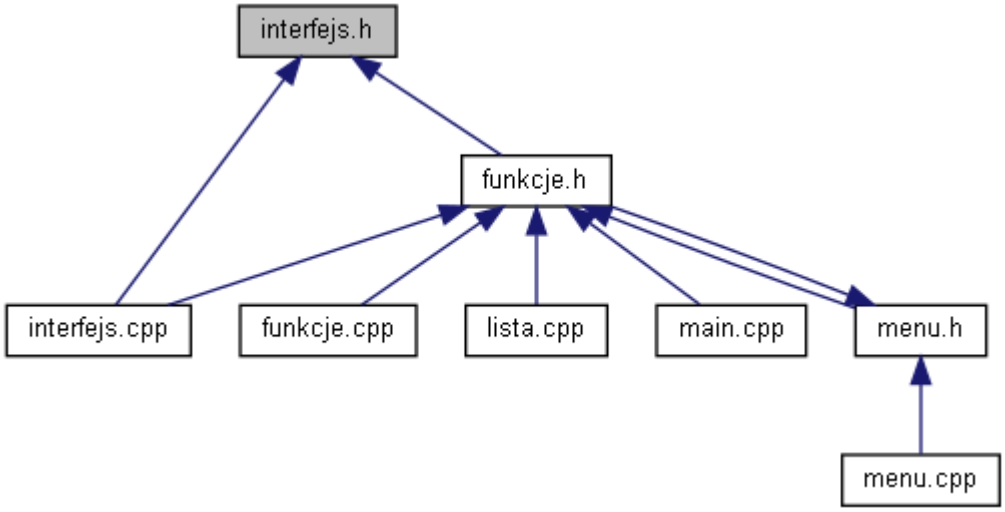
Funkcja, która cofa użytkownika do menu, które przedstawia opcje modyfikowania zmiennych. Funkcja realizuje się poprzez wyczyszczenie ekranu wiersza poleceń oraz wywołanie funkcji **modyfikujZmienne()** .

Czyszczenie ekranu.

Wywołaniewywołanie funkcji **modyfikujZmienne()** .

Dokumentacja pliku interfejs.h

Plik nagłówkowy modułu Interfejs.
Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- void **intro** ()
  Deklaracja funkcji intro.

- void **wrocDoMenu** ()
  Deklaracja funkcji, która wraca do menu głównego.

- void **wrocDoModyfikacji** ()
  Deklaracja funkcji, która wraca do wyboru opcji modyfikowania zmiennych.

- void **exit** ()
  Deklaracja funkcji, która kończy działanie programu.

Opis szczegółowy

Plik nagłówkowy modułu Interfejs.

Zawiera on deklaracje podstawowych funkcji interfejsu, które mają na celu usprawnienie poruszania się użytkownika po aplikacji

---

Dokumentacja funkcji

void exit ()

Deklaracja funkcji, która kończy działanie programu.

Deklaracja funkcji, która kończy działanie programu.

Funkcja, która kończy działanie programu.

Czyszczenie ekranu.

Zmienna a , która przechowuje znak wpisany przez użytkownika.

Pętla, która ma powtarza się dopóki użytkownik nie zdeycduje się, czy chce zakończyc działanie programu, czy nie.

void intro ()

Deklaracja funkcji intro.

Deklaracja funkcji intro.

Funkcja, która wita użytkownika w Kalkulatorze ONP.

void wrocDoMenu ()

Deklaracja funkcji, która wraca do menu głównego.

Deklaracja funkcji, która wraca do menu głównego.

Funkcja, która cofa użytkownika do menu głównego.

void wrocDoModyfikacji ()

Deklaracja funkcji, która wraca do wyboru opcji modyfikowania zmiennych.

Deklaracja funkcji, która wraca do wyboru opcji modyfikowania zmiennych.

Funkcja, która cofa użytkownika do menu, które przedstawia opcje modyfikowania zmiennych. Funkcja realizuje się poprzez wyczyszczenie ekranu wiersza poleceń oraz wywołanie funkcji **modyfikujZmienne()** .

Czyszczenie ekranu.

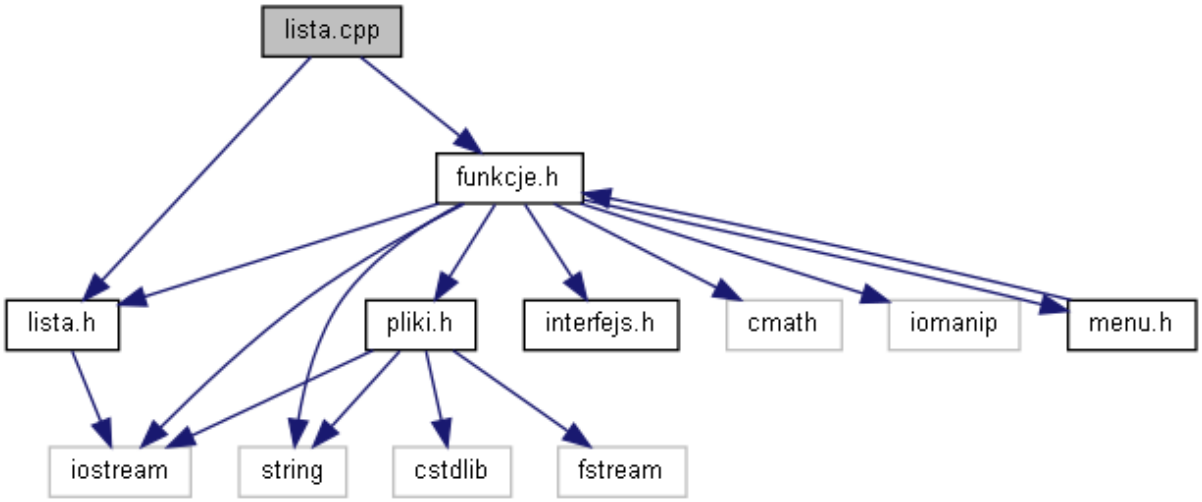
Wywołaniewywołanie funkcji **modyfikujZmienne()** .

Dokumentacja pliku lista.cpp

Plik implementacji modułu Lista.

```
#include "lista.h"
#include "funkcje.h"
```

Wykres zależności załączania dla lista.cpp:



Funkcje

- **Argument** * **stworzStos** (double x)
  Funkcja, która tworzy stos.

- void **dodajElement** (**Argument** *&glowa, double liczba)
  Funkcja, dodaje element do stosu.

- void **usunElementZeStosu** (**Argument** *&glowa)
  Funkcja, która usuwa pierwszy element ze stosu.

- void **wyczyscStos** (**Argument** *&glowa)
  Funkcja, która czyści stos.

- void **wypiszStos** (**Argument** *glowa)
  Funkcja wypisująca stos.

- **Zmienna** * **listaZmiennych** (double x)
  Funkcja, która tworzy głowę listy zmiennych.

- void **usunZmienna** (**Zmienna** *&glowa, int i)
  Funkcja usuwająca zmienną

- void **usunWszystkieZmienne** (**Zmienna** *&w)
  Funkcja usuwająca wszystkie zmienne.

- void **wypiszListeZmiennych** (**Zmienna** *w, int i)

Funkcja wypisująca listę zmiennych.

Opis szczegółowy

Plik implementacji modułu Lista.

Zawiera on definicje funkcji potrzebnych do działania na listach.

Dokumentacja funkcji

void dodajElement (Argument *& *glowa*, double *liczba*)

Funkcja, dodaje element do stosu.

Definicja funkcji, która dodaje nowy element do stosu

Parametry

**&glowa*	wskaźnik do głowy listy argumentów
liczba	wartość liczbowa nowego elementu

Zmienna* listaZmiennych (double *x*)

Funkcja, która tworzy głowę listy zmiennych.

Definicja funkcji, która alokuje pamięć na pierwszą zmienną.

Parametry

x	wartość liczbowa pierwszej zmiennej

Zwraca

glowa wskaźnik do pierwszego elementu listy

Argument* stworzStos (double *x*)

Funkcja, która tworzy stos.

Definicja funkcji, która alokuje pamięć na głowę stosu.

Zwraca

stos wskaźnik do pierwszego elementu stosu

void usunElementZeStosu (Argument *& *glowa*)

Funkcja, która usuwa pierwszy element ze stosu.

Definicja funkcji, która usuwa pierwszy argument ze stosu, czyli głowę listy dynamicznej.

Parametry

**&glowa*	wskaźnik do głowy listy argumentów

void usunWszystkieZmienne (Zmienna *& *w*)

Funkcja usuwająca wszystkie zmienne.

Definicja funkcji, która usuwa wszystkie elementy listy zmiennych.

Parametry

*&w	wskaźnik typu **Zmienna**

void usunZmienna (Zmienna *& *glowa*, int *i*)

Funkcja usuwająca zmienną

Definicja funkcji, która usuwa wybraną zmienna.

Parametry

*&w	wskaźnik typu **Zmienna**
i	nr zmiennej

void wyczyscStos (Argument *& *glowa*)

Funkcja, która czyści stos.

Definicja funkcji, która usuwa wszystkie elementy ze stosu, czyli listy dynamicznej jednokierunowej argumentów

Parametry

*&glowa	wskaźnik do głowy listy argumentów

void wypiszListeZmiennych (Zmienna * *w*, int *i*)

Funkcja wypisująca listę zmiennych.

Definicja funkcji, która wypisuje wszystkie elementy listy zmiennych.

Parametry

*w	wskaźnik typu **Zmienna**
i	liczba do numeracji kolejnych elementów listy

void wypiszStos (Argument * *glowa*)

Funkcja wypisująca stos.

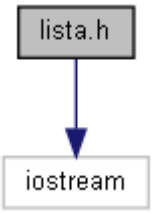
Definicja funkcji, która wypisuje listę dynamiczną jednokierunkową argumentów.

Parametry

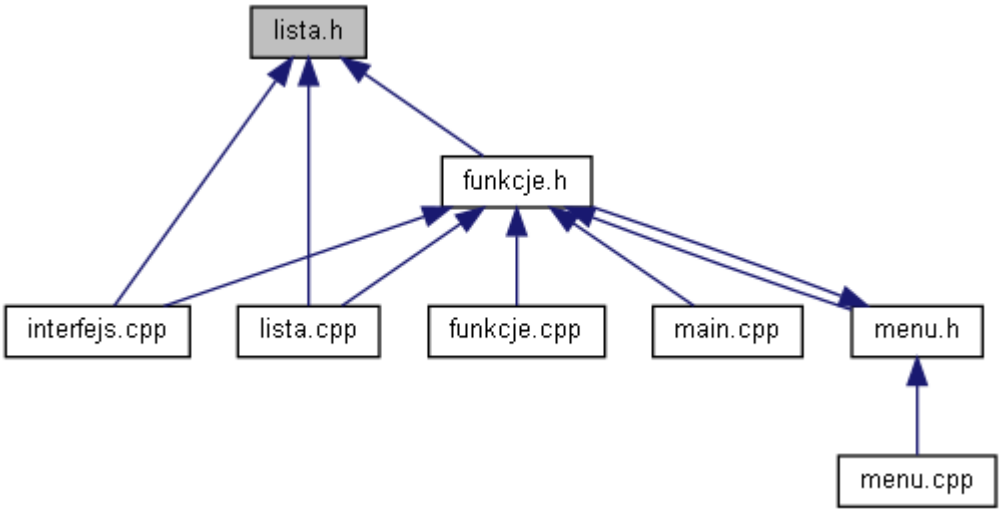
*&glowa	wskaźnik do głowy listy argumentów

Dokumentacja pliku lista.h

Plik nagłówkowy modułu Lista.
`#include <iostream>`
Wykres zależności załączania dla lista.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- struct **Argument**
  Element listy dynamicznej.
- struct **Zmienna**
  Element listy dynamicznej.

Funkcje

- void **usunZmienna** (**Zmienna** *&w, int i)
  Funkcja usuwająca zmienną

- void **usunWszystkieZmienne** (**Zmienna** *&w)
  Funkcja usuwająca wszystkie zmienne.

- void **wypiszListeZmiennych** (**Zmienna** *w, int i)
  Funkcja wypisująca listę zmiennych.

- **Argument** * **stworzStos** ()
  Funkcja, która tworzy stos.

- void **dodajElement** (**Argument** *&glowa, double liczba)
  Funkcja, dodaje element do stosu.

- void **wyczyscStos** (**Argument** *&glowa)
  Funkcja, która czyści stos.

- void **usunElementZeStosu** (**Argument** *&glowa)
  Funkcja, która usuwa pierwszy element ze stosu.

- void **wypiszStos** (**Argument** *glowa)
  Funkcja wypisująca stos.

Opis szczegółowy

Plik nagłówkowy modułu Lista.

Zawiera on deklaracje funkcji potrzebnych do działania na listach.

Dokumentacja funkcji

void dodajElement (Argument *& *glowa*, double *liczba*)

Funkcja, dodaje element do stosu.

Deklaracja funkcji, która ma dodać nowy element do stosu

Parametry

**&glowa*	wskaźnik do głowy listy argumentów
liczba	wartość liczbowa nowego elementu

Definicja funkcji, która dodaje nowy element do stosu

Parametry

**&glowa*	wskaźnik do głowy listy argumentów
liczba	wartość liczbowa nowego elementu

Argument* stworzStos ()

Funkcja, która tworzy stos.

Deklaracja funkcji, która ma zaalokować pamięć na głowę stosu.

Zwraca

stos wskaźnik do pierwszego elementu stosu

void usunElementZeStosu (Argument *& *glowa*)

Funkcja, która usuwa pierwszy element ze stosu.

Deklaracja funkcji, która ma usunąć pierwszy argument ze stosu, czyli głowę listy dynamicznej.

Parametry

&glowa	wskaźnik do głowy listy argumentów

Definicja funkcji, która usuwa pierwszy argument ze stosu, czyli głowę listy dynamicznej.

Parametry

&glowa	wskaźnik do głowy listy argumentów

void usunWszystkieZmienne (Zmienna *& *w*)

Funkcja usuwająca wszystkie zmienne.

Deklaracja funkcji, która usuwa wszystkie elementy listy zmiennych.

Parametry

*&w	wskaźnik typu **Zmienna**

Definicja funkcji, która usuwa wszystkie elementy listy zmiennych.

Parametry

*&w	wskaźnik typu **Zmienna**

void usunZmienna (Zmienna *& *glowa*, int *i*)

Funkcja usuwająca zmienną

Deklaracja funkcji, która ma ma usuwać wybraną zmienna.

Parametry

*&w	wskaźnik typu **Zmienna**
i	nr zmiennej

Definicja funkcji, która usuwa wybraną zmienna.

Parametry

*&w	wskaźnik typu **Zmienna**
i	nr zmiennej

void wyczyscStos (Argument *& *glowa*)

Funkcja, która czyści stos.

Deklaracja funkcji, która ma usunąć wszystkie elementy ze stosu, czyli listy dynamicznej jednokierunowej argumentów

Parametry

*&glowa	wskaźnik do głowy listy argumentów

Definicja funkcji, która usuwa wszystkie elementy ze stosu, czyli listy dynamicznej jednokierunowej argumentów

Parametry

*&glowa	wskaźnik do głowy listy argumentów

void wypiszListeZmiennych (Zmienna * *w*, int *i*)

Funkcja wypisująca listę zmiennych.

Deklaracja funkcji, która wypisuje wszystkie elementy listy zmiennych.

Parametry

**w*	wskaźnik typu **Zmienna**
i	liczba do numeracji kolejnych elementów listy

Definicja funkcji, która wypisuje wszystkie elementy listy zmiennych.

Parametry

**w*	wskaźnik typu **Zmienna**
i	liczba do numeracji kolejnych elementów listy

void wypiszStos (Argument * *glowa*)

Funkcja wypisująca stos.

Deklaracja funkcji, która ma wypisać listę dynamiczną jednokierunkową argumentów.

Parametry

**&glowa*	wskaźnik do głowy listy argumentów

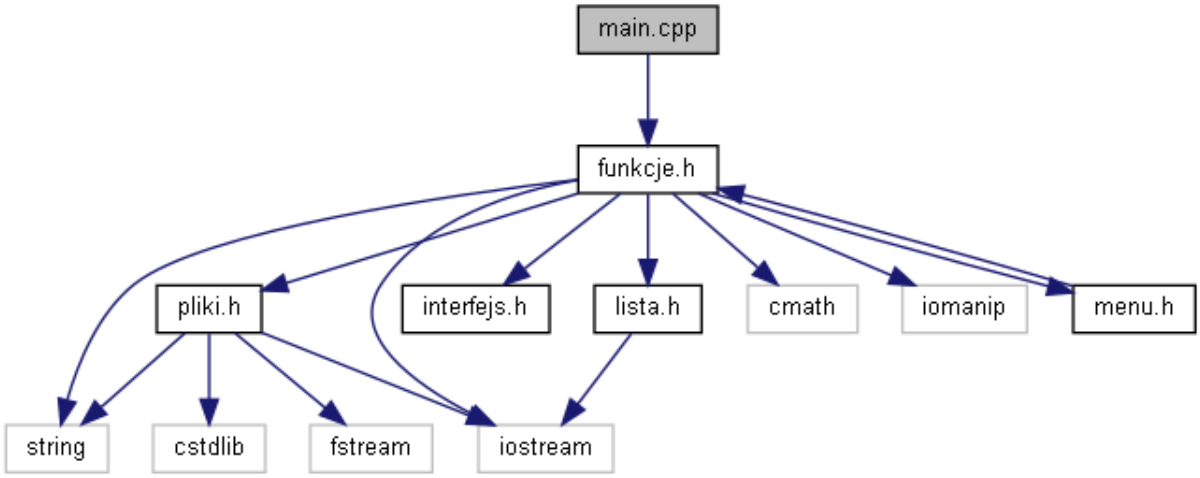
Definicja funkcji, która wypisuje listę dynamiczną jednokierunkową argumentów.

Parametry

**&glowa*	wskaźnik do głowy listy argumentów

Dokumentacja pliku main.cpp

Główny plik źródłowy.
`#include "funkcje.h"`
Wykres zależności załączania dla main.cpp:



Funkcje

- int **main** ()

Opis szczegółowy

Główny plik źródłowy.

Dokumentacja funkcji

int main ()

Intro

Funkcja, która wyświetla się tylko raz na początku. Funkcja wita użytkownika w Kalkulatorze ONP.

Menu

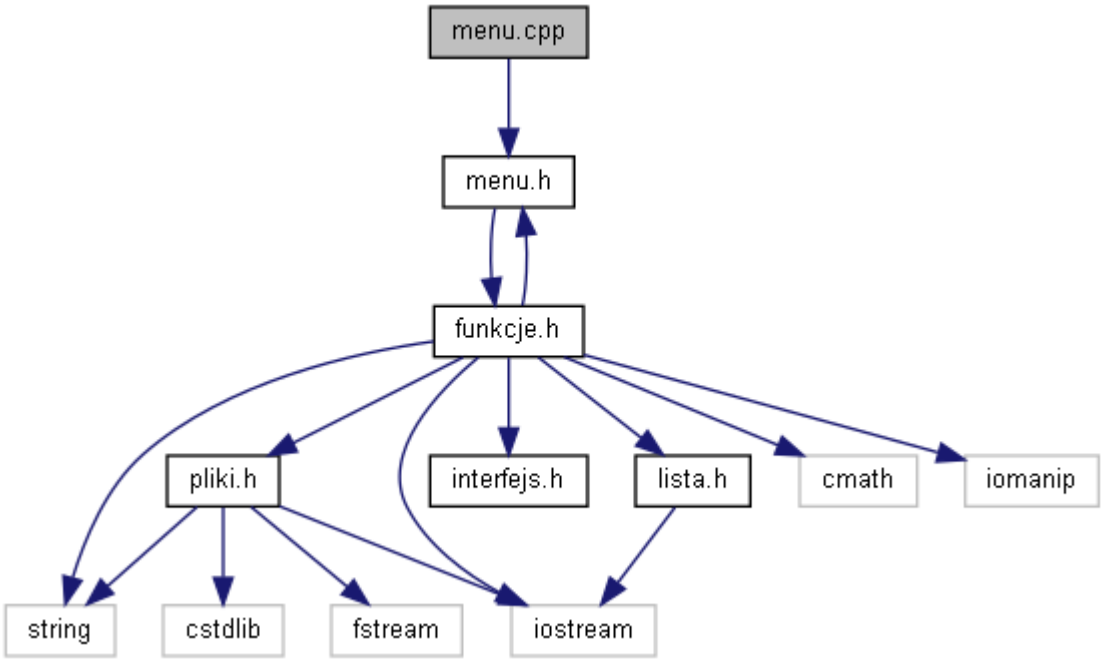
Funkcja nawigująca po programie.

Dokumentacja pliku menu.cpp

Plik impementacji modułu Menu.

```
#include "menu.h"
```

Wykres zależności załączania dla menu.cpp:



Funkcje

- void **menu** ()
  Menu.

Opis szczegółowy

Plik impementacji modułu Menu.

Dokumentacja funkcji

void menu ()

Menu.

Funkcja, która nawiguje po programie. Daje do wyboru 3 opcje:

- 1. Wprowadź lub edytuj i następnie oblicz wyrazenie.
- 2. Wprowadź lub edytuj zmienne.
- 3. Zakończ działanie programu.

Zmienna b, która będzie przechowywała napis wprowadzany przez użytkownika

Zmienna a, która pobiera pierwszy znak z napisu b

Zmienna counter ma na celu zliczanie powtórzeń pętli **do { ...} while** (...) tak aby pomóc użytkownikowi wpisanie oczekiwanej wartości zmiennej b

Pętla **do { ...} while** (...) która ma celu doprowadzenie użytkownika do wyboru jedej z trzech opcji. Pętla powtarza się do momentu wpisania napisu, którego pierwszy znak będzie równy 1, 2 lub 3.

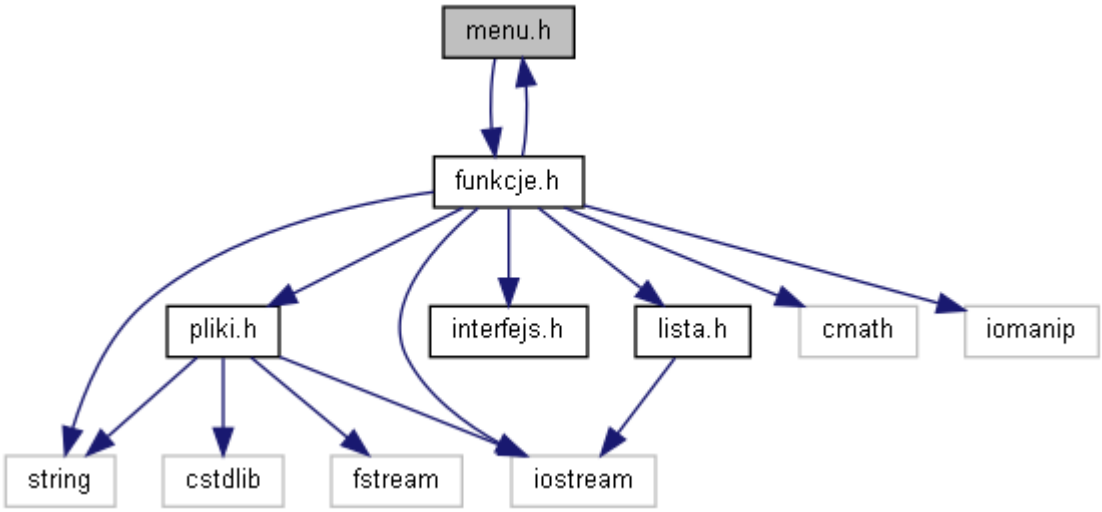
W przypadku, gdy pierwszy znak wpisanego napisu po kolejny nie jest jedną z opcji, wyświetla sie komunikat, ze wpisywana wartość jest nepoprawna.

Po poprawnym wybraniu opcji licznik powtórzeń się zeruje.

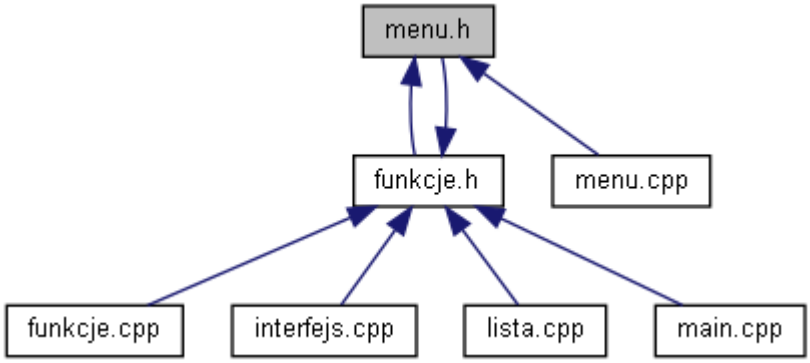
Instrukcja **switch ... case** która zależy od zmiennej a

Dokumentacja pliku menu.h

Plik nagłówkowy modułu menu.
`#include "funkcje.h"`
Wykres zależności załączania dla menu.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- void **menu** ()
  Menu.

Opis szczegółowy

Plik nagłówkowy modułu menu.

Dokumentacja funkcji

void menu ()

Menu.

Deklaracja funkcji nawigującej po programie.

Funkcja, która nawiguje po programie. Daje do wyboru 3 opcje:

- 1. Wprowadź lub edytuj i następnie oblicz wyrazenie.
- 2. Wprowadź lub edytuj zmienne.
- 3. Zakończ działanie programu.

Zmienna b, która będzie przechowywała napis wprowadzany przez użytkownika

Zmienna a, która pobiera pierwszy znak z napisu b

Zmienna counter ma na celu zliczanie powtórzeń pętli **do { ...} while** (...) tak aby pomóc użytkownikowi wpisanie oczekiwanej wartości zmiennej b

Pętla **do { ...} while** (...) która ma celu doprowadzenie użytkownika do wyboru jedej z trzech opcji. Pętla powtarza się do momentu wpisania napisu, którego pierwszy znak będzie równy 1, 2 lub 3.

W przypadku, gdy pierwszy znak wpisanego napisu po kolejny nie jest jedną z opcji, wyświetla sie komunikat, ze wpisywana wartość jest nepoprawna.

Po poprawnym wybraniu opcji licznik powtórzeń się zeruje.

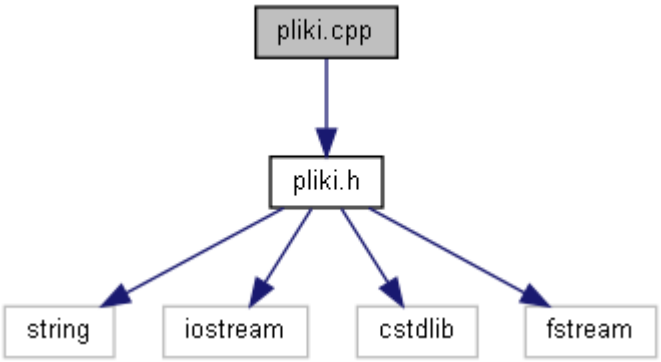
Instrukcja **switch ... case** która zależy od zmiennej a

Dokumentacja pliku pliki.cpp

Plik implementacji modułu Pliki.

```
#include "pliki.h"
```

Wykres zależności załączania dla pliki.cpp:



Funkcje

- string **pobierzZawartosc** (string **nazwa_pliku**)
  Funkcja, która pobiera zawartość pliku.

- void **usunZawartosc** (string **nazwa_pliku**)
  Funkcja, która usuwa zawartość pliku.

- void **edytujZawartosc** (string **nazwa_pliku**)
  Funkcja, która edytuje zawartość pliku.

Opis szczegółowy

Plik implementacji modułu Pliki.

Zawiera on definicje funkcji potrzebnych do działania na plikach tekstowych.

Dokumentacja funkcji

void edytujZawartosc (string *nazwa_pliku*)

Funkcja, która edytuje zawartość pliku.

Definicja funkcji, która ma edytuje zawartość pliku.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

string pobierzZawartosc (string *nazwa_pliku*)

Funkcja, która pobiera zawartość pliku.

Definicja funkcji, która pobiera i zwraca zawartość pliku w postaci napisu.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

Zwraca

zawartosc napis przechwujący zawartość pliku tekstowego

void usunZawartosc (string *nazwa_pliku*)

Funkcja, która usuwa zawartość pliku.

Definicja funkcji, która usuwa zawartość pliku.

Parametry

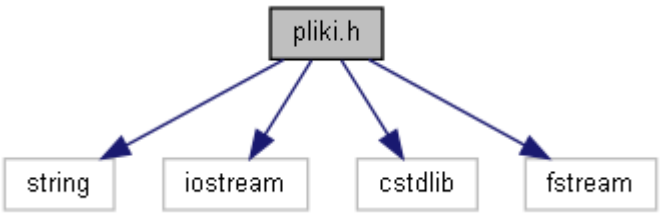
nazwa_pliku	ścieżka docelowego pliku tekstowego

Dokumentacja pliku pliki.h

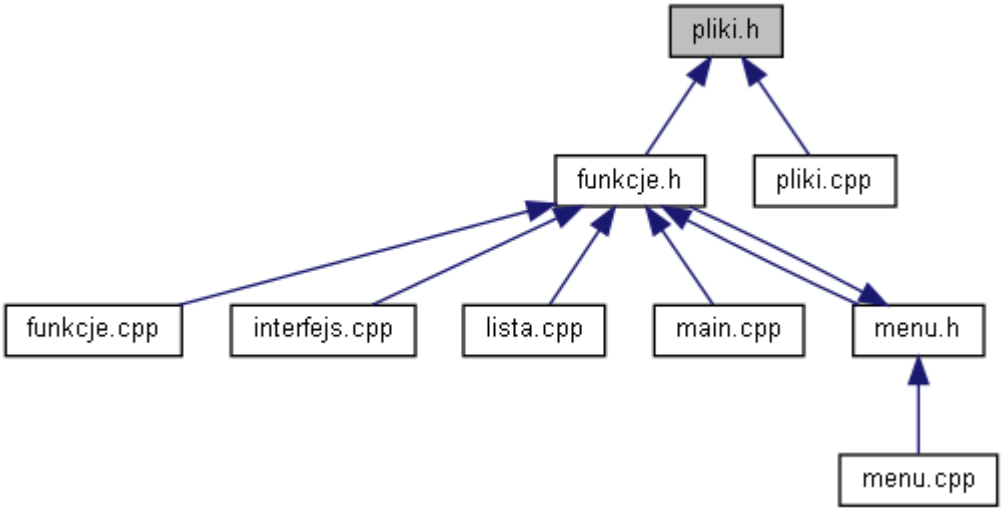
Plik nagłówkowy modułu Pliki.

```
#include <string>
#include <iostream>
#include <cstdlib>
#include <fstream>
```

Wykres zależności załączania dla pliki.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- string **pobierzZawartosc** (string **nazwa_pliku**)
  Funkcja, która pobiera zawartość pliku.

- void **usunZawartosc** (string **nazwa_pliku**)
  Funkcja, która usuwa zawartość pliku.

- void **edytujZawartosc** (string **nazwa_pliku**)
  Funkcja, która edytuje zawartość pliku.

- void **usunPlik** (string **nazwa_pliku**)
  Funkcja, która usuwa plik tekstowy.

Opis szczegółowy

Plik nagłówkowy modułu Pliki.

Zawiera on deklaracje funkcji potrzebnych do działania na plikach tekstowych.

Dokumentacja funkcji

void edytujZawartosc (string *nazwa_pliku*)

Funkcja, która edytuje zawartość pliku.

Deklaracja funkcji, która ma edytować zawartość pliku.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

Definicja funkcji, która ma edytuje zawartość pliku.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

string pobierzZawartosc (string *nazwa_pliku*)

Funkcja, która pobiera zawartość pliku.

Deklaracja funkcji, która ma pobrać i zwrócić zawartość pliku w postaci napisu.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

Zwraca

zawartosc napis przechwujący zawartość pliku tekstowego

Definicja funkcji, która pobiera i zwraca zawartość pliku w postaci napisu.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

Zwraca

zawartosc napis przechwujący zawartość pliku tekstowego

void usunPlik (string *nazwa_pliku*)

Funkcja, która usuwa plik tekstowy.

Deklaracja funkcji, która ma usunąć podany plik tekstowy

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

void usunZawartosc (string *nazwa_pliku*)

Funkcja, która usuwa zawartość pliku.

Deklaracja funkcji, która ma usunąć zawartość pliku.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

Definicja funkcji, która usuwa zawartość pliku.

Parametry

nazwa_pliku	ścieżka docelowego pliku tekstowego

Indeks

INDEX