KOMPILACJA

Zalecanym systemem do kompilacji i uruchamiania jest Linux. By skompilować program należy posiadać następujące pakiety:

- cairo
- gtk+2.0
- makefile

Kompilacja polega na wpisaniu w konsoli: „*make -f makefile*”.

JAK KORZYSTAĆ Z PMAF

Po uruchomieniu programu pojawia nam się okienko, w którym znajduję się „belka” do wpisania wzoru matematycznego oraz przycisk. Do „ładnego” wyświetlenia wzoru możemy nacisnąć ENTER lub kliknąć przycisk. Po chwili powinien nam się pojawić wzór w dolnej części okna.

Wpisywać wzory do wyświetlenia należy według kilku zasad:

1) Można stosować zmienne od ***a*** do ***z*** oraz cyfry **0**-**9**.
2) Do opisania dodawania należy użyć ' + ' np. 2 + 3, *wyrażenie* + 4.
3) Do opisania odejmowania należy użyć ' - ' np. 2 – 3, *wyrażenie* – 4.
4) Do opisania mnożenia należy użyć ' ***** ' np. 2 * 3, *wyrażenie* * 4.
5) Do opisania ułamków należy użyć ' / ' np. 2 / 3, *wyrażenie* / 4.
6) Do opisania potęgowania należy użyć ' ^' np. 2^2.
7) Do opisania indeksu dolnego należy użyć '***[zmienna]_indeks'*** np. [x]_2.

W programie obowiązują priorytety operatorów:
1) potęgowanie
2) mnożenie lub dzielenie
3) dodawanie lub odejmowanie
By narzucić swoją kolejność operatorów należy używać nawiasów '()'.

JAK DZIAŁA PMAF

Program składa się z trzech modułów:

- main.c (odpowiedzialny za tworzenie okien i odpowiednie ich pakowanie, odbieranie sygnałów (ENTER, przycisk) czy polecenie odświeżenia wzoru).
- functions.c (odpowiedzialny za wszelkie funkcje analizujące wyrażenie, odpowiednie budowanie drzewa czy jego rysowania za pomocą cairo, wyświetlanie odpowiednich błędów).
- functions.h (nagłówki funkcji i nowe typy danych).

Program korzysta z następujących funkcji:

- analyse – przekazuje tekst do funkcji analizujących, wywołuje funkcje do wyświetlania. wzoru w konsoli i okienku, poszerza okienko do wyświetlenia, jeśli jest to niezbędne.
- show_error – wyświetla daną jako parametr wiadomość o błędzie.
- operate – rozpoczyna analizę zadanego wyrażenia.
- add_operation – tworzy nowy element struktury dla danego operatora.
- add_num – tworzy nowy element struktury dla liczby.
- add_chr_with_subscript – tworzy nowy element struktury dla zmiennej z indeksem dolnym.
- add_parantheses – tworzy nowy element struktury dla wyrażenia w nawiasach.
- turn_back_chr – cofa wskaźnik, by litera była możliwa do dalszej analizy.
- read_number – odczytuje liczbę i ją zwraca.
- read_chr_extended – analizuje czy przetwarzana będzie liczba, czy zmienna.
- expression – analizuje wyrażenie składające się ze składników i '+','-'.
- term – analizuje składniki składające się z czynników i operatorów '*','/'.
- power – analizuje czynniki składające się z prostych wyrażeń i '^'.
- factor – proste wyrażenie analizuje liczby, zmienne i wyrażenia w nawiasach.
- new_malloc_element – tworzy malloc'a w gtk.
- print_on_console_with_new_line – wywołuje funkcje do wyświetlenia wyrażenia w konsoli, po czym wyświetla nową linię.
- print_on_console – wyświetla wyrażenie w konsoli.
- debug – wyświetlanie różnych wiadomości w konsoli przy debuggowaniu.
- expose_event_callback – funkcja potrzebna dla sygnału odświeżania cairo.
- do_drawing – rysuje wzór w oknie.
- max – zwraca większą z liczb.
- do_xy – nadaje elementom struktury współrzędne x i y do wyświetlenia.
- how_many_digits – zwraca ilość cyfr w liczbie.
- create_pixbuf – funkcja do tworzenia pixbuf'a (potrzebna dla ikony programu).

Program ma zdefiniowane dwie struktury:

- ELEMENT (zawiera zmienne boolowskie definiujące czy jest to liczba, czy zmienna, czy operator, czy może nawias, współrzędne obiektów i ich długości oraz same obiekty).
- WINDOW_AND_SEARCH_ENTRY (zawiera obiekty GTK potrzebne do komunikacji pomiędzy modułami i do przekazania ich do odpowiednich funkcji).

Program korzysta też z kilku zmiennych globalnych niezbędnych do zachowywania pewnych wyników obliczeń pomiędzy funkcjami.