

Plot - how the program works

The idea of this program is to take the entered string as an algebraic expression and adopted it to a form that allows systematic operations of simple operators (+, -, *, /) on two numbers after:

- checking a number of points that I arbitrarily assumed for myself, such as the impossibility of entering more than 70 characters of an algebraic expression, more than 6 digits of the integer part of a number and the same number of fractional parts, writing the same mathematical function in an expression more than twice (e.g. $f(x) = \sin(x) + \sin(3 \times \ln(x/2)) - 1/\sin(x)$ is impossible for the program to execute due to the occurrence of **sin** three times), etc.,
- checking if the number entered by the user is a real number (complex numbers are not accepted),
- detecting a notation that must be a mathematical function and checking its syntax according to the syntax given in the table of functions to which it is always access. E.g. the square root is **sq()** and not **sqrt()***,
- removing unnecessary spaces entered by the user,
- accepting only two notations 'e' and 'pi' as constant numbers; 'pi' is visually replaced with π ,
- taking control over the entered numbers (the limits of reasonableness of the size of numbers, the syntax of a number should be with a decimal point, etc.),
- calculating of the value (for a given 'x') in the innermost parentheses,
- calculating the values of mathematical functions of a specific number of α (or two specific numbers for **mod(α , β)** or **pow(α , β)**) when α and β are concrete (fixed or already calculated) numbers.

The order of the above last points does not have to be observed at all.

The program checks if the parentheses () are paired with each other, but the program does not append the missing) to the end of the expression.

The vast majority of these points do not apply to the 'polynomial', for which the domain of the function is automatically given as $X \in (-\infty; +\infty)$.

Introduction

This program doesn't pretend to be perfect. Even sending this code to IT companies while looking for a programmer job, didn't give me an opportunity to be employed by one of them. It was the same with XYZ.bas (three-dimensional function graphics - you can find it on this website) several years earlier.

Since 1998, when I created the program**, the ASCII table has been changed, both English and Polish code pages. Hence the strange markings in the program:

- In the meantime, mathematical signs became impoverished: I replaced the previously available symbol ∞ with « or », ϵ with ε (they are look similar each other), hence, for example, the writing $X \in (-\epsilon; +\epsilon)$ instead of the previous $X \in (-\infty; +\infty)$ ***. The approximation symbol (\sim above the = sign) disappeared and only the \approx sign remained on the English code page.
- The English code page has more Greek characters than any other one. This is the result of savings on special national characters of many countries (e.g. there are 18 additional Polish characters). Hence the availability of such characters used in the English version of this program such as α , $\&$ and π . Be aware of this if you are from a non-English speaking country and would like to change this program to one that words and sentences would be understood by children in your country.
- The numbers of graphic characters were exchanged, both in Polish and English ASCII. Although the ASCII of these characters in both Polish and English are the same, their position in the table has been changed since 1998). If your code page is neither 437 (English) nor 852 (Polish), please, be aware that your boxes in the program could require changes (graphic characters such as \parallel — \perp | \lrcorner \lceil etc. to build a box).

But once upon a time I defined my own characters on a 'Commodore' computer. Similarly, in the 'Word Perfect' (the predecessor of *Microsoft Word*), I defined Polish characters without any problems. It was a 'piece of cake' to do that.

Other shortcomings of the program are described below.

Yet I used this program often. Here you can easily find parts of the code that you would like to implement into your own programs.

For non-English reader: You can find entering numbers with a decimal comma instead of a decimal point - please see 'Typing numbers with the decimal comma' below.

I have kept some standards like:

- [Esc] for exit from an active window,
- [F1] for an informative 'help' window,
- [PgUp] and [PgDn] for scrolling windows down and up,
- [Tab] but the same time arrow keys [←] and [→] or [↑] and [↓] work in all required windows,
- If you do not know what to do, just press [Enter].

Don't worry that the **plot.cpp** program in the notebook contains strange (unexpected) characters, even though it was saved in the notebook with the 'Encoding: UTF-8' option. The code copied to the Turbo C++ environment should show the correct characters.

Similarly, the possible uncontrolled scatter of the lines (e.g. larger indents than expected) after copying **plot.cpp** from notebook to the **Turbo C++** environment is also irrelevant.

* The notation **sq** is as mnemonic as **sqrt**, to which **sq** is converted in the program. My assumption was that it could be possible to introduce notations such as **cosine(α)** and **Kosinus(α)** instead of **cos(α)**, into which **cosine** and **Kosinus** would be converted. I also planned to define a logarithm (of a positive number **β**) with an arbitrary positive base **α** : **log(α, β)** as **ln(β)/ln(α)** or **lg(β)/lg(α)**.

** I tried to write such a program in LOGO on ZX Spectrum in 1988, but it quickly crashed due to small RAM (16kB).

*** I accept such as notation because there are no strictly established standards in mathematics. For example, in the past it was customary to denote **ln** as a natural logarithm, **lg** as a decimal logarithm, **lb** as a binary logarithm and **log** as a logarithm with any base, and the absence of the base meant base 10. Now academic books do not follow that concept at all.

Borland Turbo C++ installation

Turbo C++ installation is described below on this website: *Plot (Wykres funkcji) --> Borland Turbo C++ Installation and programs description.*

Functions

Functions and their positions in the program

Function prototypes and their lines in the program

void main (void)	235
void introduction (void);	252
void function_type_selection (void);	387
void the_largest_window (void);	2205
void polynomial_choosing (void);	454
void remove_spaces_and_analyze (void);	628
void correct_notation_of_the_number (void);	689
void check_number_size (void);	737
void free_form_function_choosing (void);	796
void question_about_notation (void);	1839
void operators_and_math_func_table (void);	893
void introduction_graphics (void);	1930
void parameters_selection (void);	2103
void OX_axis_unit_length (void);	2230
void OY_axis_unit_length (void);	2308
void background_pattern_number (void);	2427
void bgr_patterns_and_colors_window (void);	3078
void background_color_number (void);	2500
void coor_system_axes_color_number (void);	2575
void function_graph_color_number (void);	2656

void routine_introduction(void);	2736
void preliminary_analysis_of_chars(void);	2766
void no_possibility_of_correction(void);	2907
void function_graph_background(void);	3576
void graph_of_the_function(void);	3733
Error report	
void this_is_not_a_real_number(void);	666
void string_is_too_long(void);	2936
void remove_spaces(void);	2914
void this_is_not_a_natural_number(void);	2956
void out_of_range_number(void);	2976
void the_same_color_is_not_allowed(void);	2995
void clear_error_report(void);	3017
void clear_info(void);	3043
void error_found(void);	4848
Exit the program and help	
void whether_to_exit_the_program(void);	3148
void contradictory_sentences(int hop_count);	3227
void whether_to_exit_the_program_gr(void);	3263
void help_non_gr(void);	3346
void help_gr(void);	3521
Functions and their types for an "free-form function"	
void read_the_string_and_analyze_it(void);	986
void correct_algebraic_expression(void);	1805
void is_there_any_argument(void);	1762
void inner_parenth_operations_test(void);	3973
void inner_parenth_operations(void);	4398
void remove_only_parentheses(void);	4369
void insert_number_into_the_string(void);	4297
float result_of_a_simple_operation(float aa, float bb);	4471
Other basic (apart from read_the_string_and_analyze_it) elements of "free-form functions"	
void free_form_function(void);	3942
void math_function_calculation(void);	4098
int is_it_a_lack_of_calculations(void);	4266
The "menu" window after the graph is completed	
void function_menu(void);	4507
void function_analysis(void);	4695
void window_of_function_analysis(void);	4840

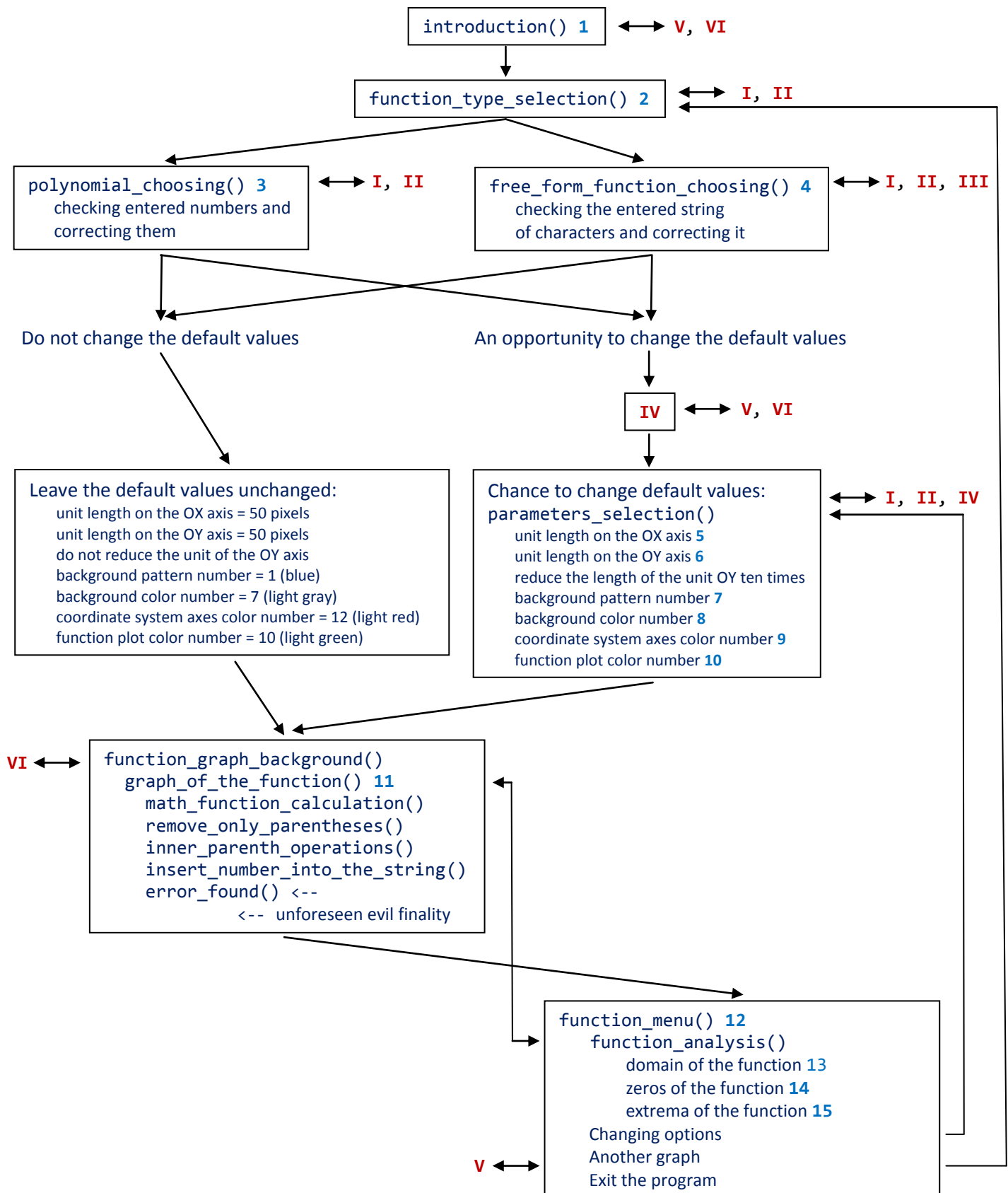
Functions - flow chart

Symbols used below:

whether_to_exit_the_program() = I	exiting the program from non-graphic screen window
help_non_gr() = II	informational 'help' window on a non-graphic screen
operators_and_math_func_table() = III	preview of operators and mathematical functions syntax
introduction_graphics() = IV	lista opcji do zmian z podglądem do tablicy barw
whether_to_exit_the_program_gr() = V	exiting the program from graphic screen window
help_gr() = VI	informational 'help' window on a graphic screen

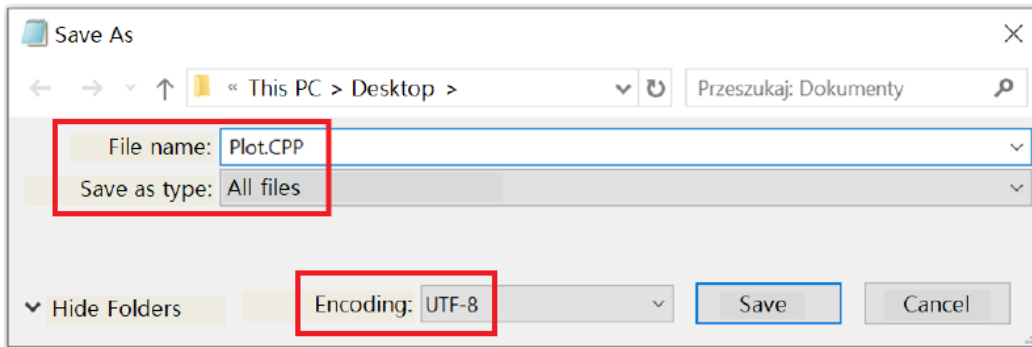
Roman numerals (above) and Arabic numerals (below) will be needed to mark the screenshots presented under the process flow diagram.

A very schematic (but as clear as possible) flow of the process

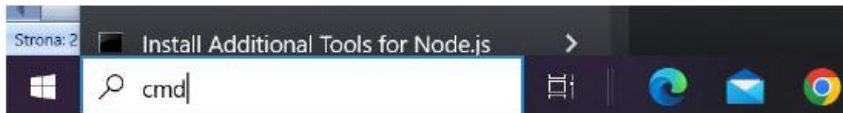


How to install *Plot.cpp* in the *TurboC3* environment.

1. Open the 'Function Plot' file containing the code (*Making a two-dimensional plot of a function - English*),
2. Open Microsoft notebook,
3. 'copy' all the contents of the 'Function Plot' to the notebook,
4. Save (Save as) the notebook file with the following parameters on the desktop:



5. Copy **Plot.cpp** from the desktop to **C:\TURBOC3\Projects\Plot**
 - a) Open **cmd** (*Command Prompt*)



- b) Go down to the *root directory* with the command **CD/**
- c) In **C:\TURBOC3**, create a **Projects** directory and a **Plot** directory in it:

```
C:\Users\Leszek>cd\  
C:\>  
C:\>cd TurboC3  
C:\TURBOC3>  
C:\TURBOC3>md Projects  
C:\TURBOC3>cd Projects  
C:\TURBOC3\Projects>  
C:\TURBOC3\Projects>md Plot  
C:\TURBOC3\Projects>cd Plot  
C:\TURBOC3\Projects\Plot>
```

- d) Copy **Plot.cpp** from the panel to the directory **C:\TURBOC3\Projects\Plot>**

C:\TURBOC3\Projects\Plot>copy C:\Users\ (Your identifier) \Desktop\Plot.CPP

Example:

```
C:\TURBOC3\Projects\Plot>copy C:\Users\Leszek\Desktop\Plot.CPP
```

If you want to copy the program from the Turbo C++ environment back to the panel, type from any level in **cmd**:

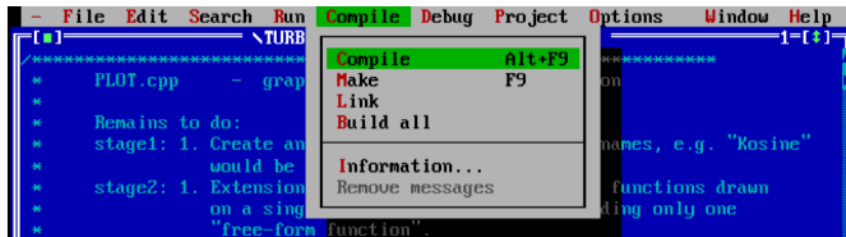
copy C:\TURBOC3\Projects\Plot\Plot.CPP C:\Users\ (Your identifier) \Desktop\Plot.CPP

Example:

```
C:\Users\Leszek>copy C:\TURBOC3\Projects\Plot\Plot.CPP C:\Users\Leszek\Desktop\Plot.CPP
```

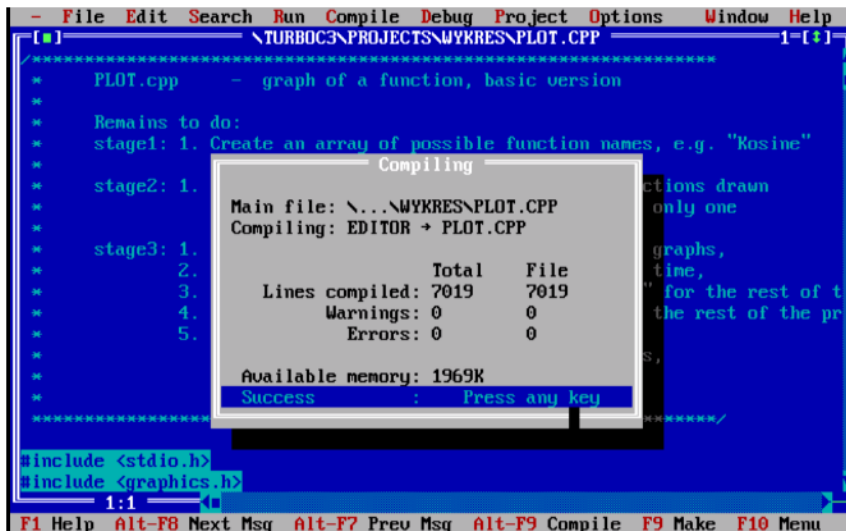
Surfing the program - step by step

Compiling the program ([Alt]+C --> [Enter] or [F10] --> Compile --> Compile --> [Enter] or [Alt]+[F9]).



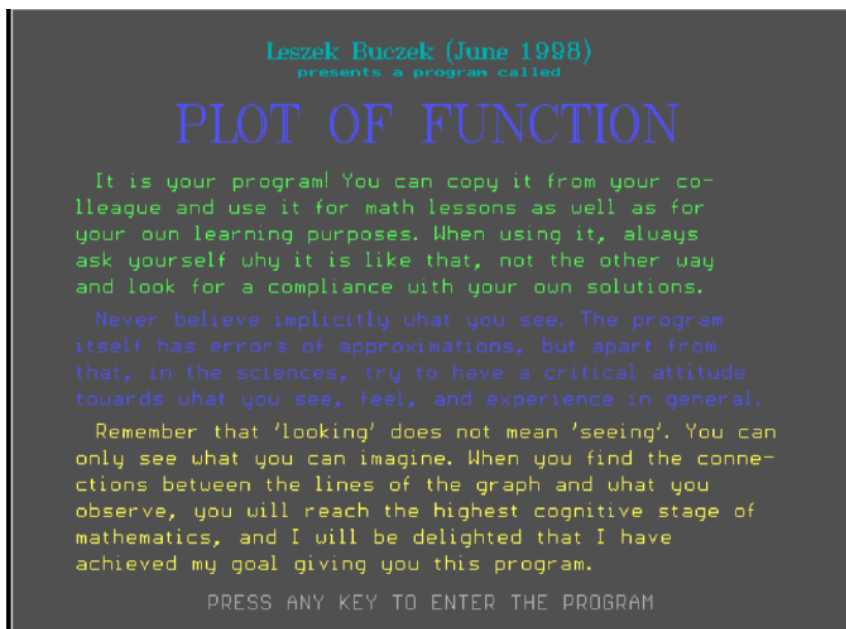
4928 program lines.

Together with 'include' files - about 7000.



If you have about 30 compilation errors, you probably did not include 'graphics' to the program. Please see: ***Little bit more about navigation in the Borland Turbo C++ v. 3.0 environment, i.e. other important Menu elements*** below.

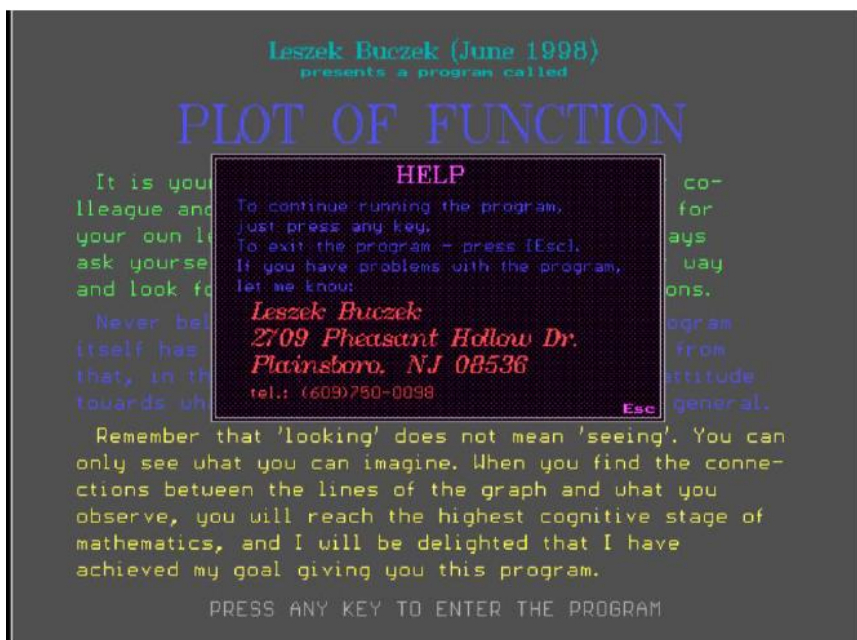
Running the program ([Alt]+R --> [Enter] or [F10] --> Run --> Run --> [Enter] or [Ctrl]+[F9]).



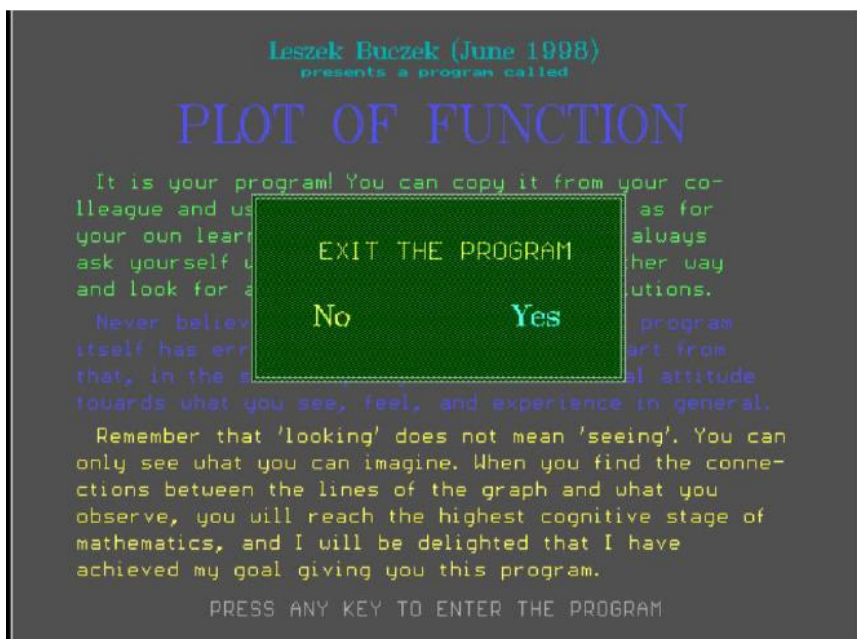
From almost every screen, the following options are available:

- 'Help' by pressing the [F1] function key, which is standard in programming (see getch() function). The 'Help' window is exited by pressing [Esc] - also the standard.
- 'Exit the program' by pressing the [Esc] key, which is also the standard in programming.

[F1] (VI) Operation key: [Esc]



[Esc] (v) Operation keys: [→],[←],[Tab].



Jumping between 'No' and 'Yes' can be done either with the arrow keys [→] and [←] or [Tab].

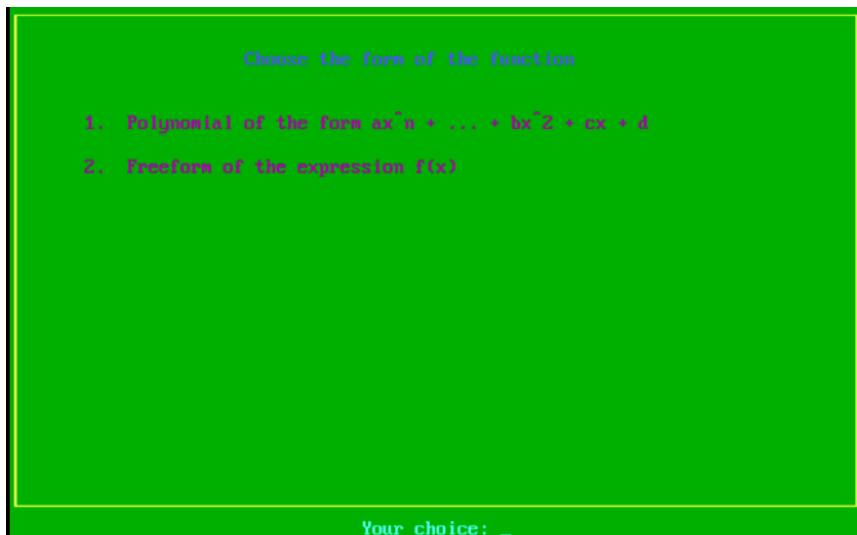
Exit the 'HELP' windows by pressing the [Esc] key.

[Enter] gives access to specifics (if you don't know what to do, press [Enter]) divided into sections here:

Sections:

1. Selecting a type of function: 'Polynomial' or 'Free-form of an expression' (2)
 - a. Polynomial form (3)
 - b. Free-form of an expression (4)
2. Changing the parameters of a function plot
3. Function plot menu
4. Example of operations with constant numbers: 'e' and 'pi'

Section 1. Selecting a type of function: 'Polynomial' or 'Free-form of an expression' (2)



From this level, you can, of course, call both 'HELP' - [F1], this time of a non-graphical window - and choose the option of 'EXIT THE PROGRAM' - [Esc], here is of a non-graphical window (after exiting the introduction() function/module, void introduction(void), the graphic was closed - with the command closegraph();)

'HELP' of non-graphical window. Five windows (the first page is below) - use the [PgDn], [PgUp] and [Esc] keys. (II)



'EXIT THE PROGRAM' of non-graphical window (I) :



Notice the changing sentences at the bottom of the screen. It seems to be unnecessary but...
 ...every educational program should be as interactive as possible - this is just an example of many elements of the program interaction with the user's activity.

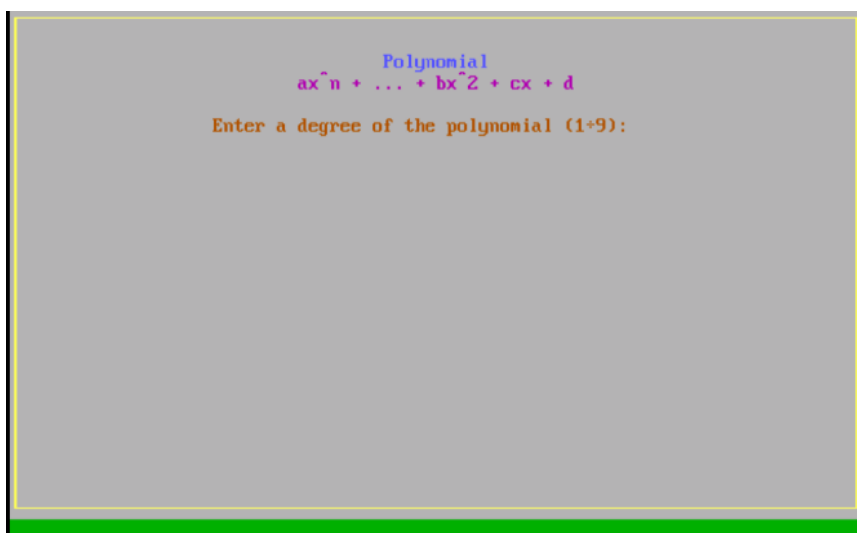
Some kind of interaction is on the 'HELP': There are 6 different 'HELP' screens/pages in non-graphic mode - the appropriate page appears at the appropriate stage of the program. However, from any page that just appears, it is possible to scroll through all other pages.

Now, we return to the choice:

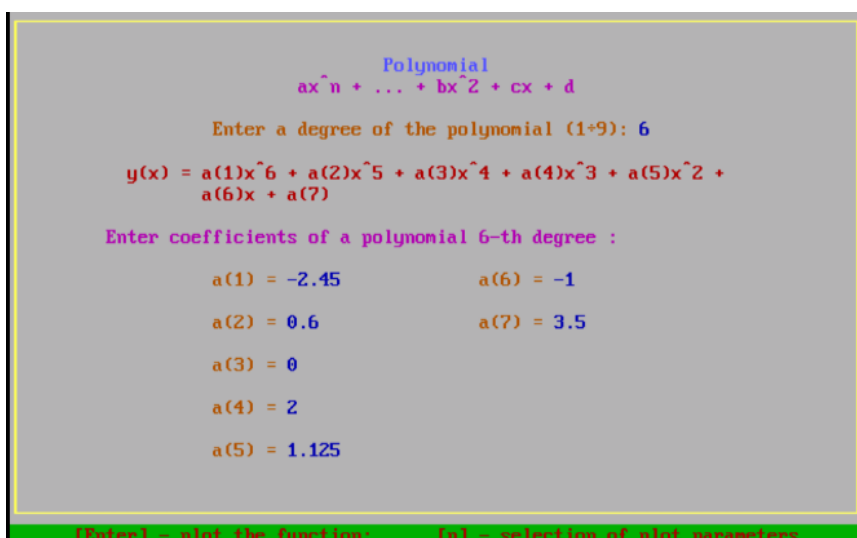
- either a 'polynomial'
- or a 'free-form function'.

And let it now be **1** (i.e. a polynomial). **[Enter]** is not needed here.

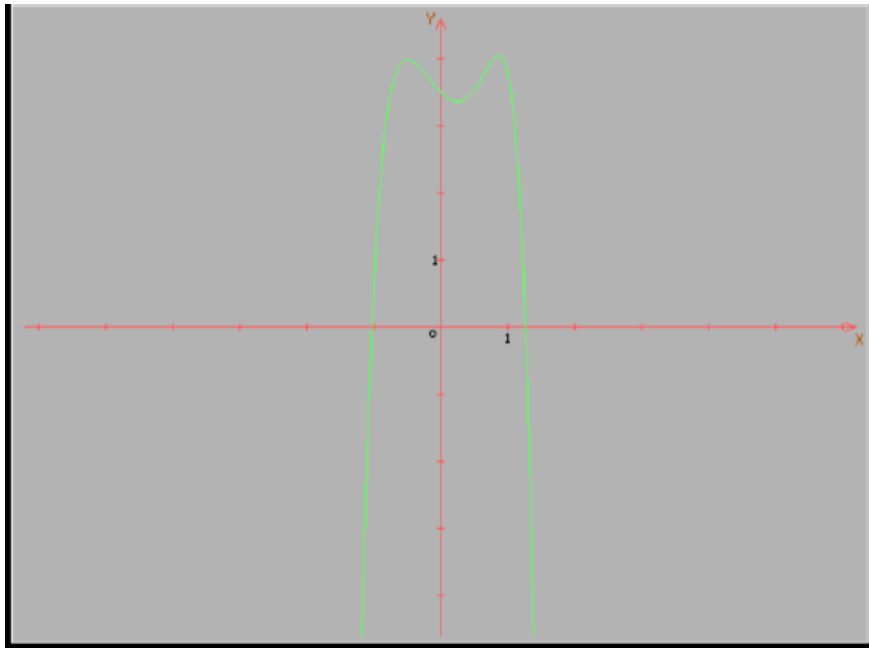
Section 1. a. Polynomial form (3)



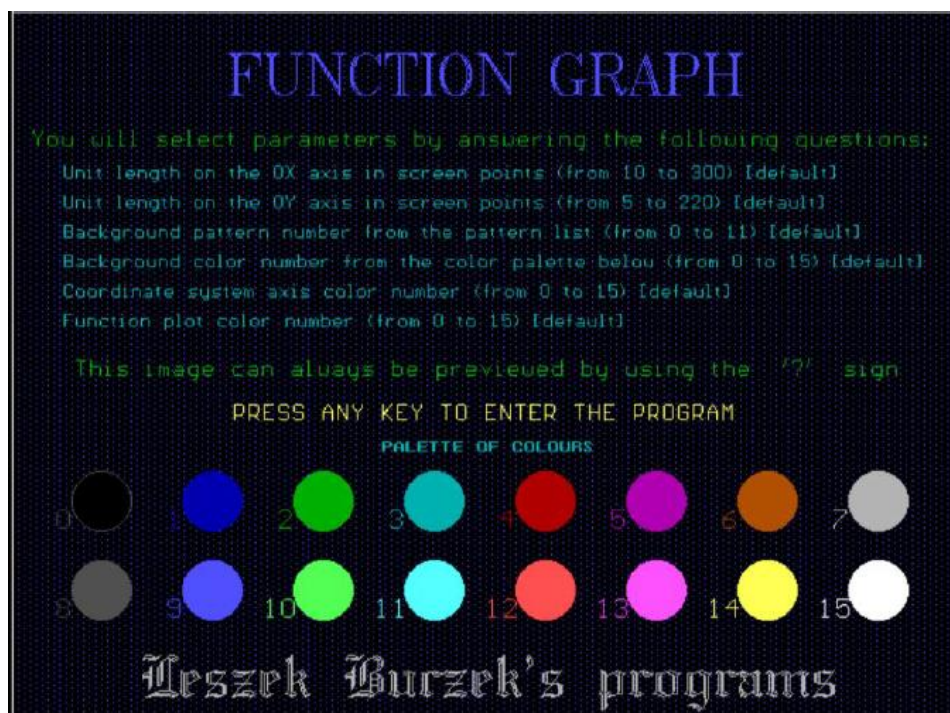
It is possible to enter a polynomial up to the ninth degree - here we enter its coefficients in real numbers. E.g.:



After pressing [**Enter**], we immediately get the function graph with its optional parameters unchanged. **(11)**

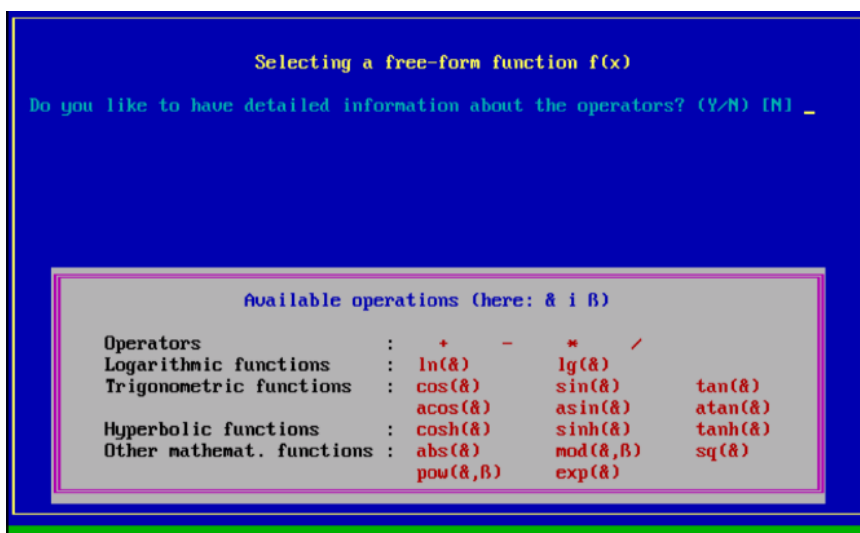


If we press [**p**] instead of [**Enter**], we will enter the level of selecting a number of parameters of the function graph with the initial image as below (so it is possible to change the parameters) (**IV**):

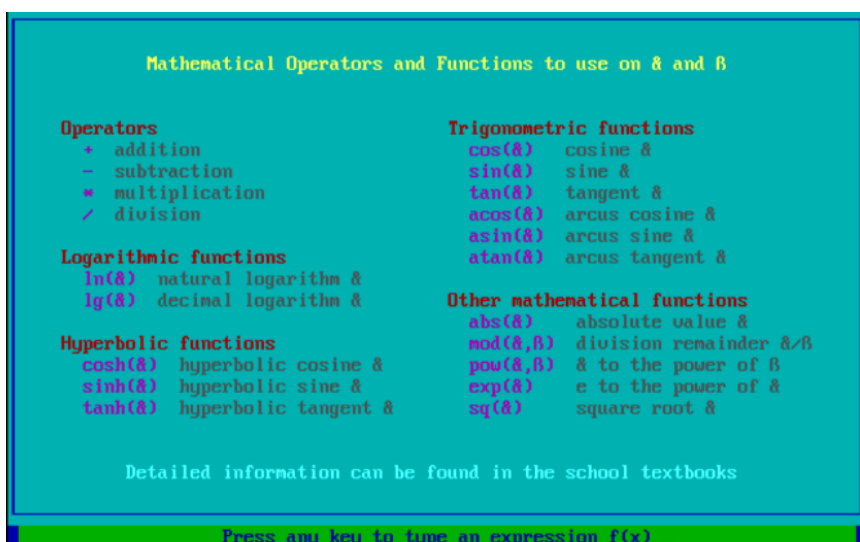


For further information on how to change the plot/graph parameters, see below: **Section 2. Changing the parameters of a function plot.**

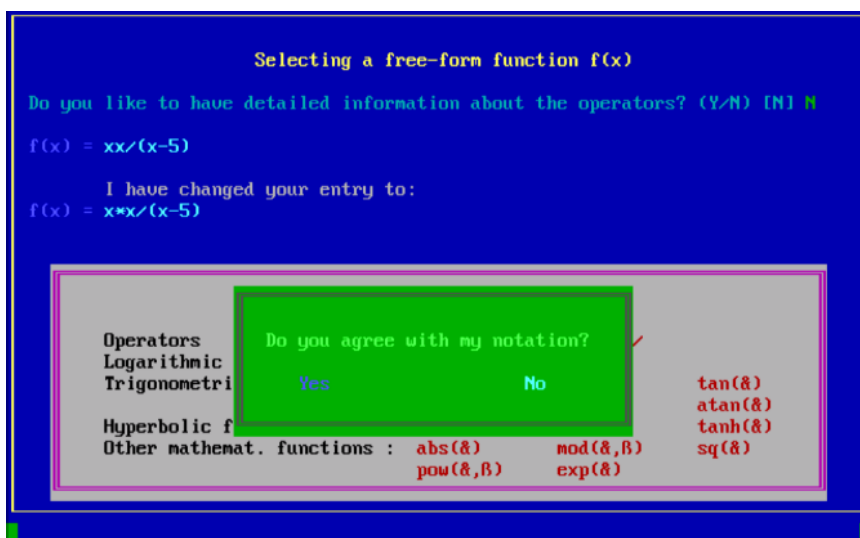
Section 1. b. Free-form of an expression (4)



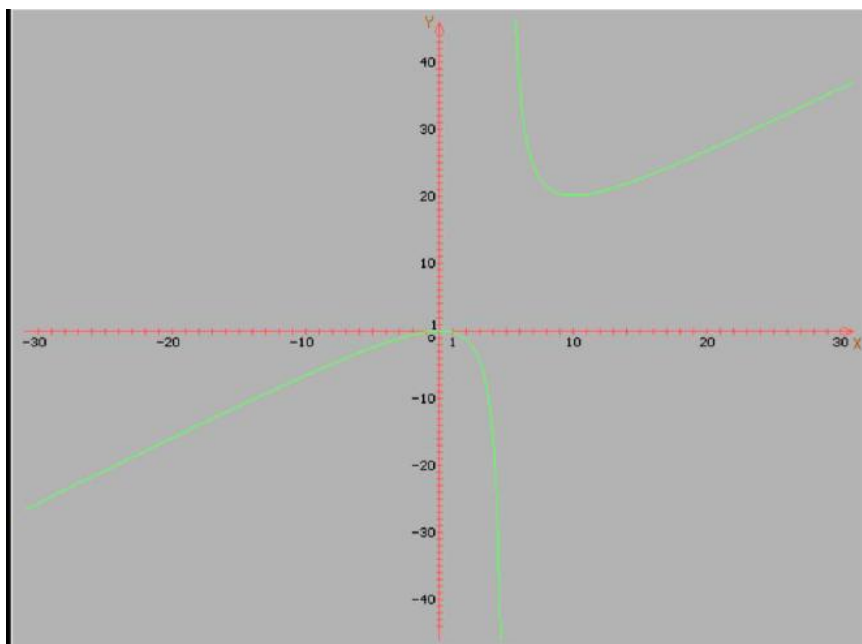
The initial query for detailed syntaxes of operators and functions is optional: Pressing [Enter] skips this screen, [y] or [Y] takes you to it:



Since I have Polish code page only and do not like to change it, I used & instead of α only to make above shown screen shots. The code PLOT.CPP has ASCII number 224 (it should be α on your computer) in each place of &.



And after changing the OX and OY unit lengths to make them as short as possible (11):



(13)

```

FUNCTION ANALYSIS
Domain of the function

X = (?;4.90) u (5.10;?)

Note: '(' may turn out to be '[' and
')' ']''. A '?' means that the graph
begins or ends with a number that
already belongs to the function domain.
Page Down < Esc

```

(14)

```

FUNCTION ANALYSIS
Zeros of the function

y = 0 for X1 = 0.00

Page Down < Page Up < Esc

```

(15)

```

FUNCTION ANALYSIS
Extrema of the function

Max: y = -0.00 for X = 0.00
Min: y = 20.00 for X = 10.00

Page Up < Esc

```

Section 2. Changing the parameters of a function plot

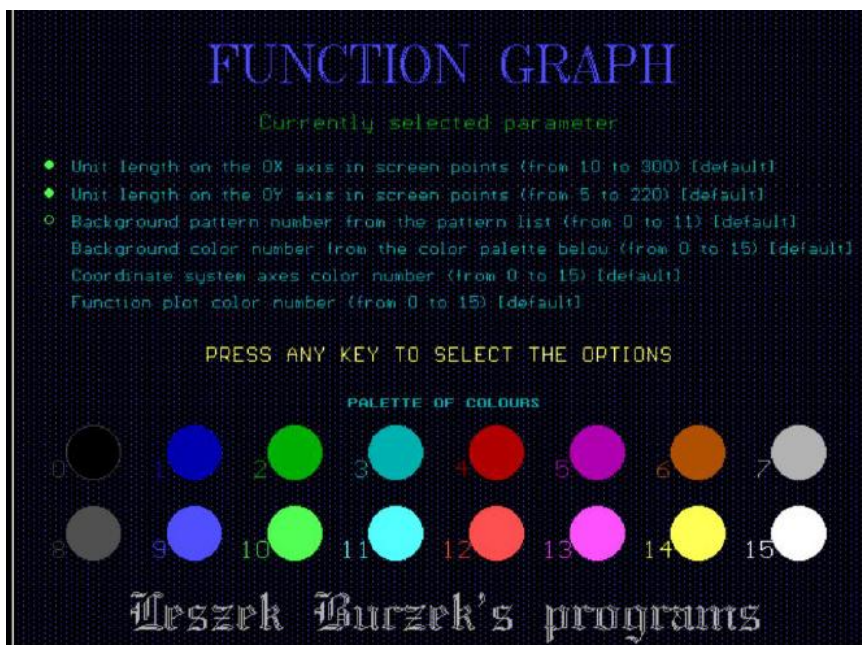
You can accept optional values for each option by pressing [Enter] (5) or ...



... change some of them (6):

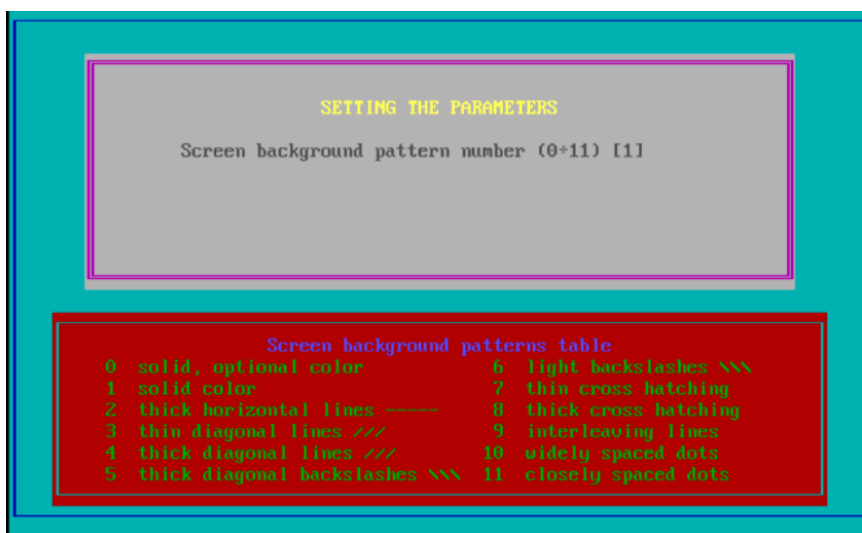


From each such screen, there is a preview of the main screen of parameter changes - **[Shift]+[?]** .
 Additionally the screen shows where we are - at what particular option - doing changes (**IV**):

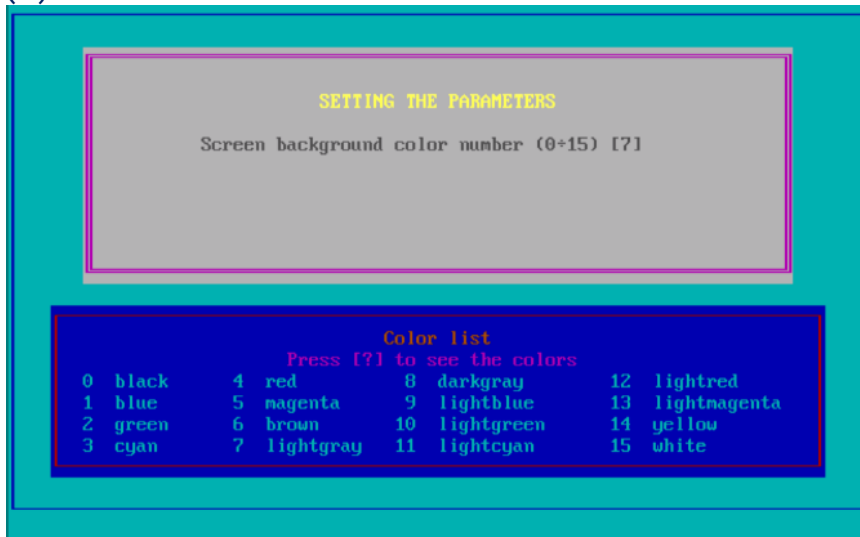


[Enter]

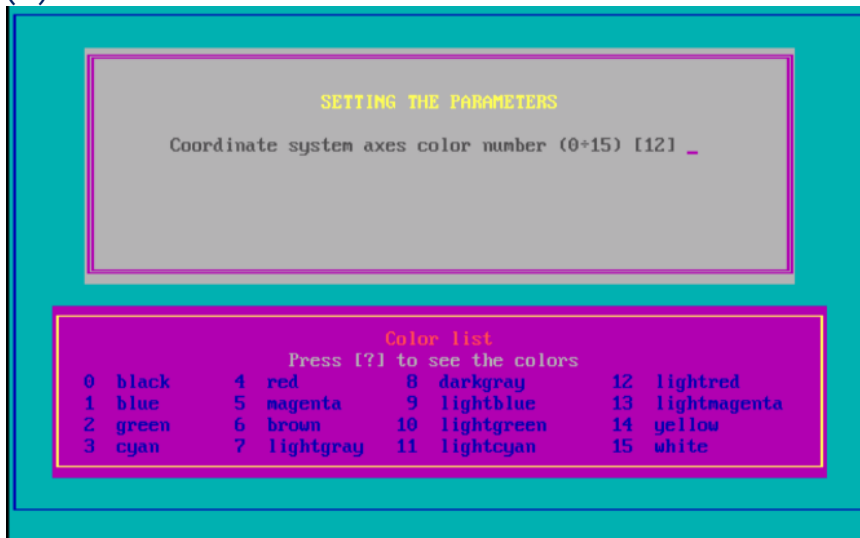
And the same with other parameters (**7**):



(8)



(9)



(10)

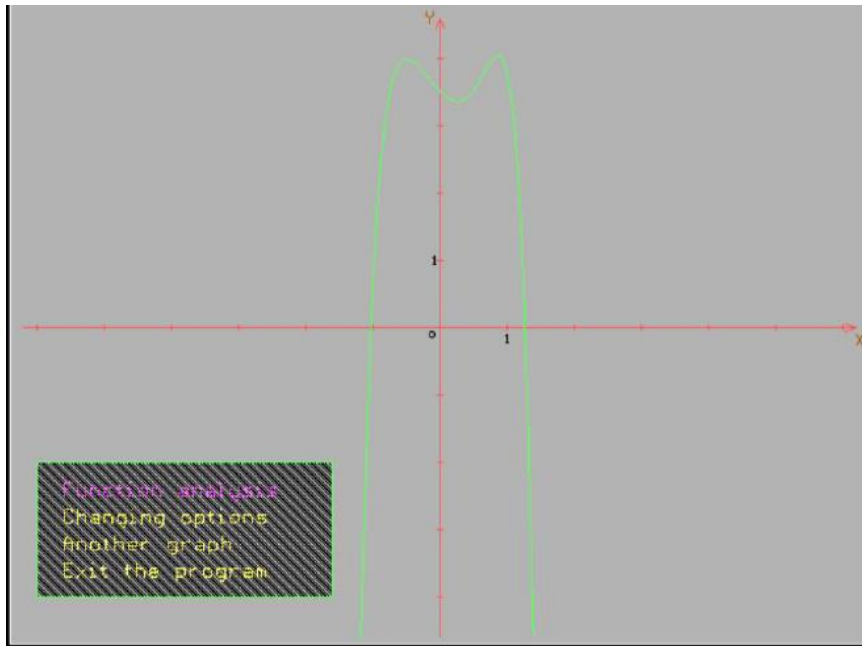


and finally:

Press any key to see the function graph

Section 3. Function plot menu

Pressing [Enter] on the function plot launches the 'Plot Menu' (12).



The jump between these four options is enable by using the [↓] and [↑] keys, and those jumps have ability to 'scroll' the menu ('Exit the program' and [↓] gives the 'Function Analysis' again), just like any option in the Turbo C++ editor menu.

Function Analysis

It contains three elements:

- i. Domain of the function
- ii. Zeros of the function
- iii. Extrema of the function

The domain of the function for 'polynomial' is automatically given as $X \in (-\infty; +\infty)$, on the screen $X \in (-\ll; +\gg)$ for the above-mentioned reasons.

The remaining outcomes are the result of a process during additional calculations leading to the preparation of a graph, and only for the X and Y values that fit on the screen. Therefore, these values should be treated as approximate ones. To get a better approximation of these numbers, change your plotting options:

Unit length on the OX axis in screen points (from 10 to 300)

and/or

Unit length on the OY axis in screen points (from 5 to 220)

So for e.g. $\sin(x)/x$ is obtained $X = (?; -0.02) \cup (0.02; ?)$. Sign ? replaces here $-\infty$ and $+\infty$ (in practice $-\ll$ and $+\gg$) because how can the program know what is happening off the screen? Changing the 'unit length on the OX axis in screen points' from 50 to 300 gives $X = (?; -0.00) \cup (0.00; ?)$.

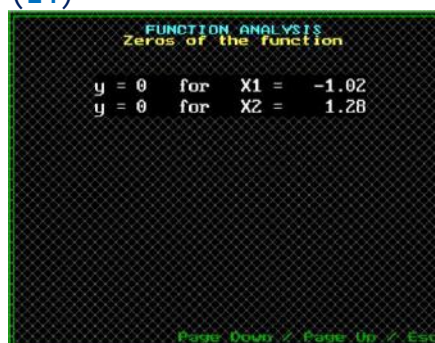
Numbers are given to two decimal places.

For the case of the drawn function, this is:

(13)



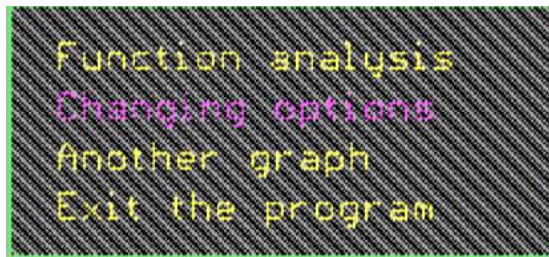
(14)



(15)

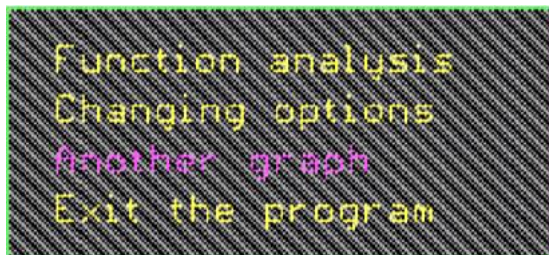


Changing options



Leads to **Section 2. Changing the parameters of a function plot**, except that its initial window does not appear, but it is available from each invoked plot parameter for possible change.

Another graph



Leads to **Section 1. Selecting a type of function: 'Polynomial' or 'Free-form of an expression' (2)**.

Exit the program



Takes you to a box from which you can either exit the program or cancel the program exiting process:



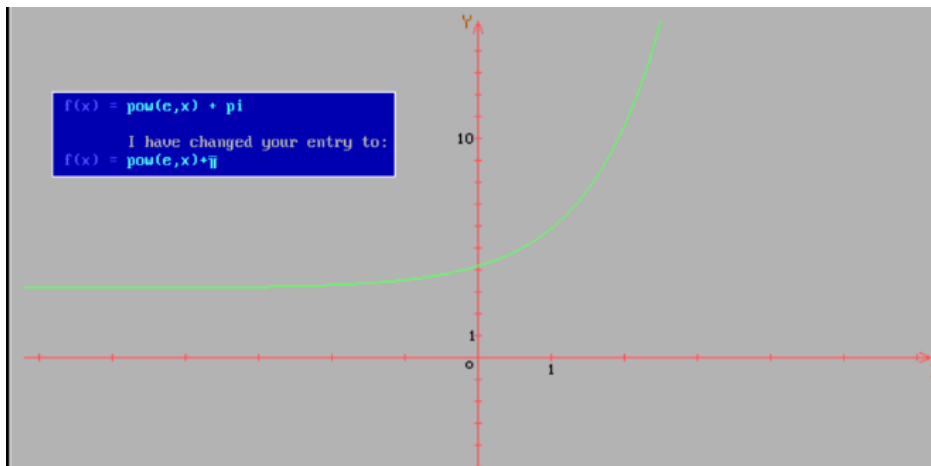
Section 4. Example of operations with constant numbers: 'e' and 'pi'

The Euler number can be written in an expression as **e**. The program will convert **e** to 2.718282 . At the same time, it will leave the symbol **e** in the visual notation of the expression.

The number **π** can be written as **pi**. The program will change this entry to 3.141593. At the same time, it will leave the symbol **π** in the visual notation of the expression.

An example of typing Euler's number and **π** at the same time is below. This function is

$f(x) = e^x + \pi$ As seen in the plot, $f(x) = e^x$ is moved up by 3.14 .



Since I have Polish code page only and do not like to change it, I used π instead of π only to make above shown screen shot. The code **PLOT.CPP** has ASCII number 227 (it should be π on your computer) in each place of π .

Some parts of how the program works that you may not understand

1. Waiting for a key to be pressed

From the definition of the **getch()** function, the following keys were used, the numbers of which were entered into the program:

#define KEY_F1	59
#define KEY_UP	72
#define KEY_PGUP	73
#define KEY_LEFT	75
#define KEY_RIGHT	77
#define KEY_DOWN	80
#define KEY_PGDN	81

Except that from the ASCII table:

Enter	13
Esc	27

2. "There is no space in the memory to store the image"

If you get an error...:

```
Error (n): There is no space in the memory to store the image
Its size = required number bytes is too large
Free memory is only number available bytes
```

where,

n - is the next informational number; it allows pointed a specific error location in the code for similar displayed information,

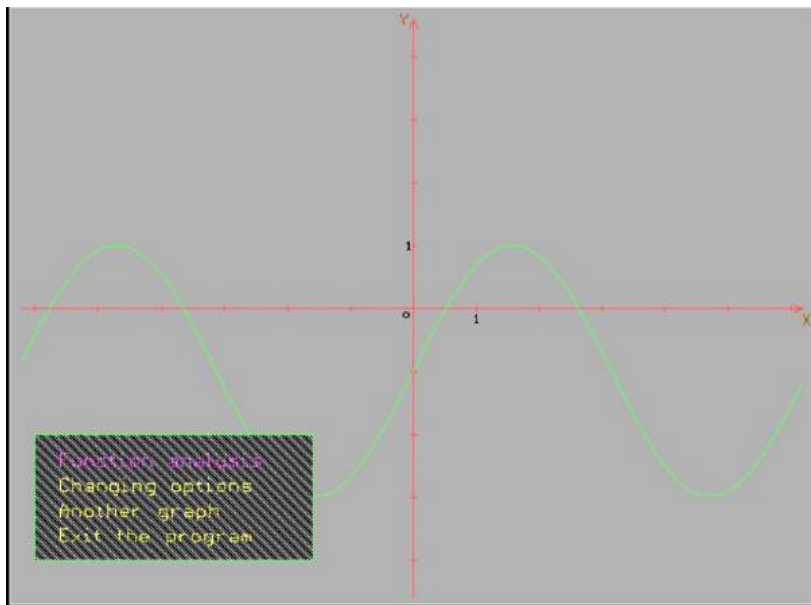
required number - is the specific number of bytes required for the image to appear on the screen,

number available - is the number of bytes in the program's operation memory.

In order to see the image on the screen (and therefore to the entire program to run), the following condition must be met:

$$\text{required number} \leq \text{number available}$$

When an attempt to call '*Function Analysis*'...

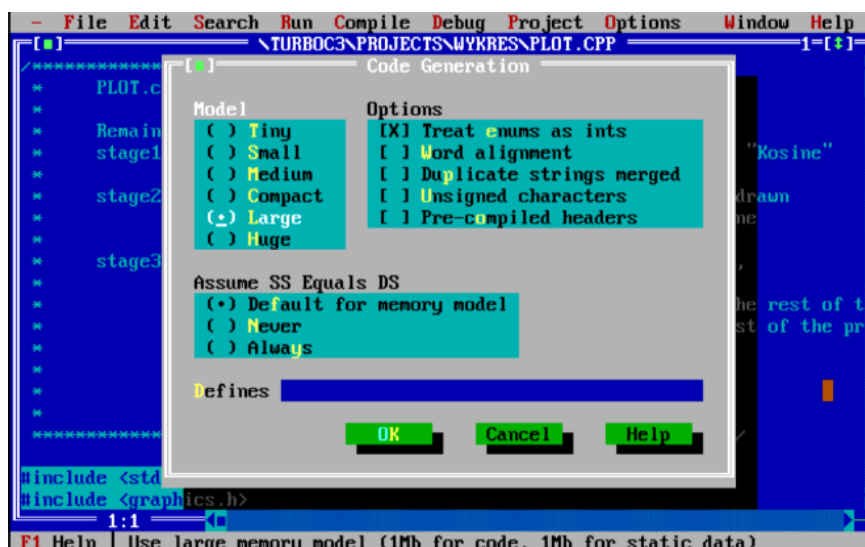
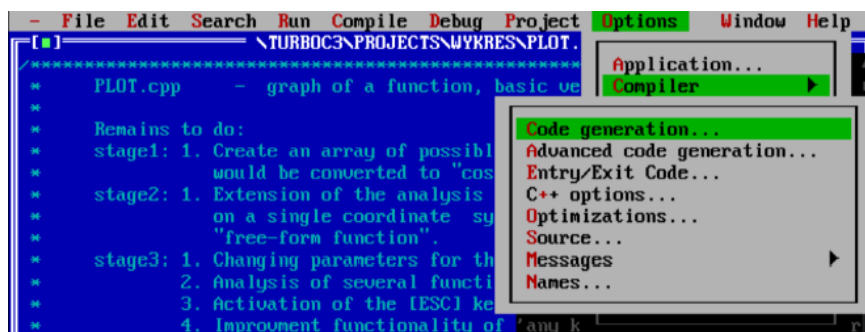


... results in an error:

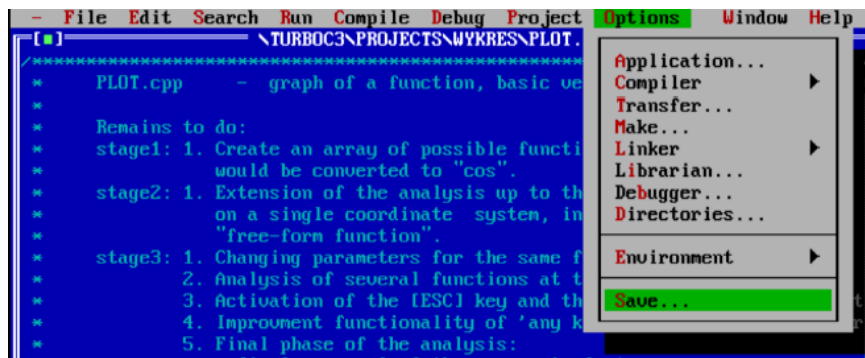
```
Error (7): There is no space in the memory to store the image
Its size = 44898 bytes is too large
Free memory is only 36160 bytes
_
```

You probably have a memory **Model** that is too small. It needs to be increased.

Then: **Option** --> **Compiler** --> **Code generation...** --> (change 'Model' to 'Large', even 'Compact' would be enough)



Do not forget to **SAVE** that option:



3. Some interactive sentences in the program

Except of 'HELP' (both graphic and non-graphic modes) and set of sentences accompanying 'EXIT THE PROGRAM' (non-graphic mode), the following tips and/or warning may appear while running the program:

"Graphics error (module '*name of a module, e.g. introduction*'): '*errorcode*'"

"Press any key to enter the editor : "

<-- there are 3 different module names (so also positions in the code) because graphics is launched 3 times

Possible information appearing at the bottom of the screen:

```
"          WRONG !  You have a choice of either "1" or "2"          "
```

```
"          Choose one of the two options (either 1 or 2)          "
```

```
"          WRONG !  You have a choice of digits 1, 2, 3, 4, 5, 6, 7, 8 and 9          "
```

```
"          Choose one of nine options (from 1 to 9)          "
```

```
"          Too many characters. [ENTER] - accept, [<--] - delete the digits.          "
```

```
"          A number can be up to 14 characters long. Spaces are allowed.          "
```

```
"          [Enter] - plot the function;          [p] - selection of plot parameters          "
```

```
"          WRONG !  It must be a real number in decimal notation here.          "
```

```
"          Examples:  3,  4.55,  .61,  0.61,  -.61,  -0.61,  -123.45,  +123.45          "
```

```
"          I did not accept!  The acceptable upper limit of a number is 999999.          "
```

```
"          Even MILLION (1000000) is too big number for me.          "
```

```
"          I did not accept!  The acceptable lower limit of a number is -999999.          "
```

```
"          The number MINUS MILLION (-1000000) and below are too small numbers to me.          "
```

```
"Attention:I have limited the number by cutting it to 6 digits after the period"
```

```
"          Press any key to type an expression f(x) _          "
```

```
"          Enter any algebraic expression here.          "
```

```
"          Example:  4abs(sin(2.5pi/cos(x2tanh(3.2x))))-pow(e,-x/pi)          "
```

```
"          Too many digits in a number.          "
```

```
"Number cannot have more than 6 digits in both the integer and fractional parts"
```

```
"          I cannot handle more characters.          "
```

```
"          You can only enter up to 70 chars. [ENTER] - accept, [<--] - remove chars.          "
```

```
"          An unacceptable mark has been found.  Correct the expression typing.          "
```

```
"          The operation requires two numbers.  Correct the expression.          "
```

```
"          Error in writing operators (+, -, *, /).  Correct the expression.          "
```

```
"          Error in placing parentheses.  Correct the expression.          "
```

```
"          The "ln()" function can only be used twice.  Include this in the expression.          "
```

```
"          The "lg()" function can only be used twice.  Include this in the expression.          "
```

```
"          The "cos()" function can only be used twice.  Include this in the expression.          "
```

```
"          The "cosh()" function can only be used twice.  Include this in the expression.          "
```

```
"          The "sin()" function can only be used twice.  Include this in the expression.          "
```


" The "sinh()" function can only be used twice. Include this in the expression."

" The "sq()" function can only be used twice. Include this in the expression. "

" The "tan()" function can only be used twice. Include this in the expression. "

" The "tanh()" function can only be used twice. Include this in the expression."

" The "acos()" function can only be used twice. Include this in the expression."

" The "asin()" function can only be used twice. Include this in the expression."

" The "atan()" function can only be used twice. Include this in the expression."

" The "abs()" function can only be used twice. Include this in the expression. "

" The "mod()" function can only be used twice. Include this in the expression. "

" The "pow()" function can only be used twice. Include this in the expression. "

" The "exp()" function can only be used twice. Include this in the expression. "

"A function is expected but its notation is incorrect. Type correct expression."

" Something is wrong with the commas. Remember "." replaces "," in a number. "

" Every function requires an argument. Check the table and correct any errors. "

" The "mod()" and "pow()" functions must have two arguments. Correct errors. "

" [Enter] - function graph; [Any other key] - selection of the graph parameters"

"Type the expression again, but in a different way so I will know what you mean"

" [Enter] - function graph; [Any other key] - selection of the graph parameters"

" There is no correction option except removing the last character [<--] "

Parameter selection sentences:

"Currently selected parameter" <-- points to it

"Press any key to see the function graph"

"Press any key to select the next parameter"

"Preview [?]"

"Such a large number of characters is unacceptable"

"Correct the data"

"The entered value is not a natural number"

"Correct the data"

"The number must be in a range [10;300]"

"Correct the data"

"The number must be in a range [5;220]"

"Correct the data"

"The number must be in a range [0;11]"

"Correct the data"

"The number must be in a range [0;15]"

"Correct the data"

"The color of the axes is the same as the background color"

"Correct the data"

"The color of the graph is the same as the background color"

"Correct the data"

Sentences about the graph of a function:

"Not a single graph point appeared in the visible area."

"A certain interval of the graph of a function requires a complex plane." <-- this sentence should be taken with a pinch of salt.

"Note: '(' may turn out to be '[' and ')' ']''. A '?' means that the graph begins or ends with a number that "already belongs to the function domain."

"I did not detect any graph points belonging to the domain of the function within the range of the ordinate axes seen on the screen."

"I did not detect any zeros in the range seen on the screen."

"I did not detect any extrema in the range seen on the screen."

Finally, if everything else does not allow to make a plot, the following message will shown up:



Sentences regarding the ultimate impossibility of making a plot ("UNEXPECTED ERROR"):

"I'm having trouble with parentheses. Their number or position is not correct."

"I expected a function, but I did not find it."

"I expected an operator, but I did not find it."

"I encountered incomprehensible operator between numbers."

4. Understanding graphics

a) Resolution managed by Turbo C++. Pixel numbers in brackets.



b) Colors - they can be declared by a number or name (capital letters)

0 - BLACK	8 - DARKGRAY
1 - BLUE	9 - LIGHTBLUE
2 - GREEN	10 - LIGHTGREEN
3 - CYAN	11 - LIGHTCYAN
4 - RED	12 - LIGHTRED
5 - MAGENTA	13 - LIGHTMAGENTA
6 - BROWN	14 - YELLOW
7 - LIGHTGRAY	15 - WHITE

c) Line Style and Thickness Names

Line style	Thickness
-----	-----
SOLID_LINE	NORM_WIDTH

DOTTED_LINE	
CENTER_LINE	THICK_WIDTH
DASHED_LINE	
USERBIT_LINE	

d) Line Style Patterns

SOLID_LINE = 0
DOTTED_LINE = 1
CENTER_LINE = 2
DASHED_LINE = 3

e) Pattern names

Values/names	Causing filling with
EMPTY_FILL	Background Color
SOLID_FILL	Solid Color
LINE_FILL	Horizontal Lines -----
LTSLASH_FILL	Thin diagonal lines /////
SLASH_FILL	Thick diagonal lines /////
BKSLASH_FILL	Thick diagonal backslashes \\\\\\\
LTBKSLASH_FILL	Light backslashes \\\\\\\
HATCH_FILL	Thin cross hatching
XHATCH_FILL	Thick cross hatching
INTERLEAVE_FILL	Interleaving lines
WIDE_DOT_FILL	Widely spaced dots
CLOSE_DOT_FILL	Closely spaced dots

f) Text styles

To set the values for the text characteristics: **settextstyle(font, direction, character size);**

Font	Direction	Character Size
DEFAULT_FONT	HORIZ_DIR <-- Left to right	1 = Default (normal)
TRIPLEX_FONT	VERT_DIR <-- Bottom to top	2 = Double Size
SMALL_FONT		3 = Triple Size
SANS_SERIF_FONT		4 = 4 times the normal
GOTHIC_FONT		5 = 5 times the normal
SCRIPT_FONT		...
SIMPLEX_FONT		10 = 10 times the normal
TRIPLEX_SCR_FONT		
COMPLEX_FONT		
EUROPEAN_FONT		
BOLD_FONT		

g) Text Justification

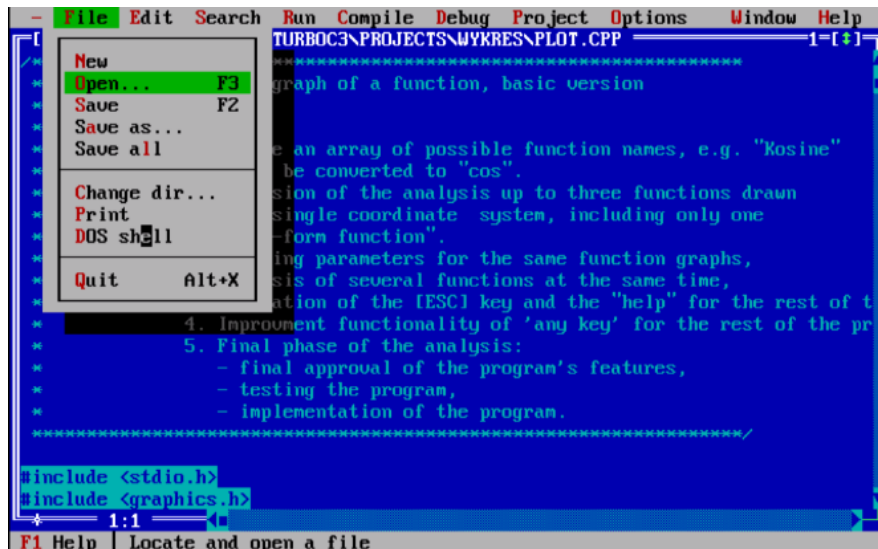
To set the way that text is located around the point specified, use the command **settextjustify(horizontal, vertical);**

Horizontal	Vertical
LEFT_TEXT	TOP_TEXT
CENTER_TEXT	BOTTOM_TEXT
RIGHT_TEXT	

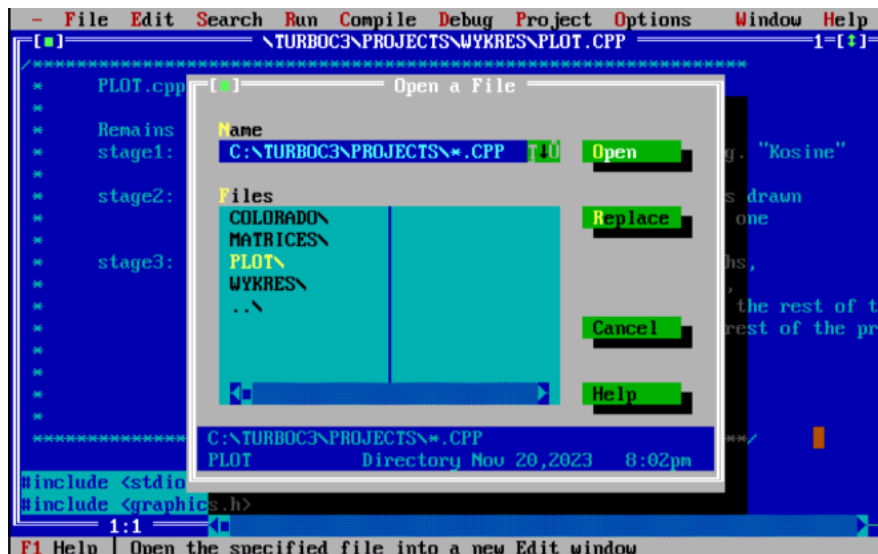
Little bit more about navigation in the Borland Turbo C++ v. 3.0 environment, i.e. other important Menu elements.

You can reach the Menu by pressing the function key [F10] or [Alt]+*appropriate letter* or [Alt]+[F] and using the arrow [→] to select the option you are looking for.

File



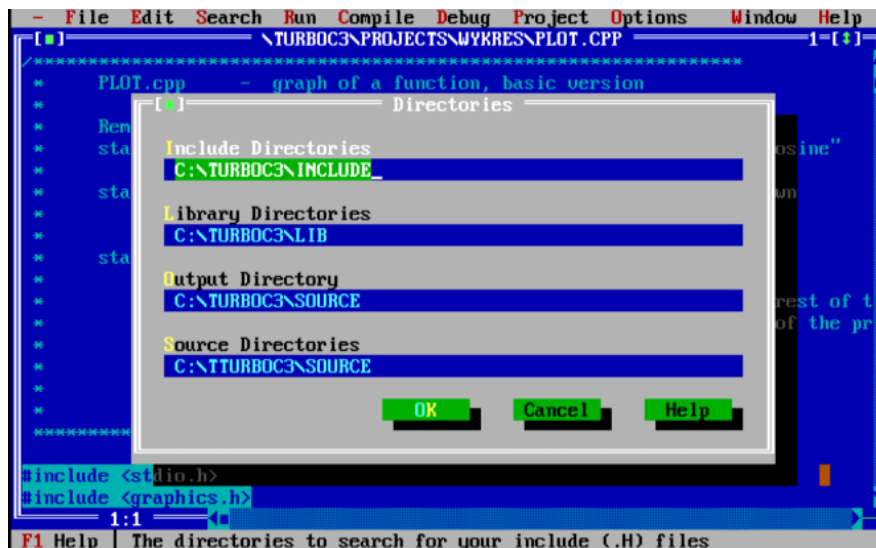
File --> Open



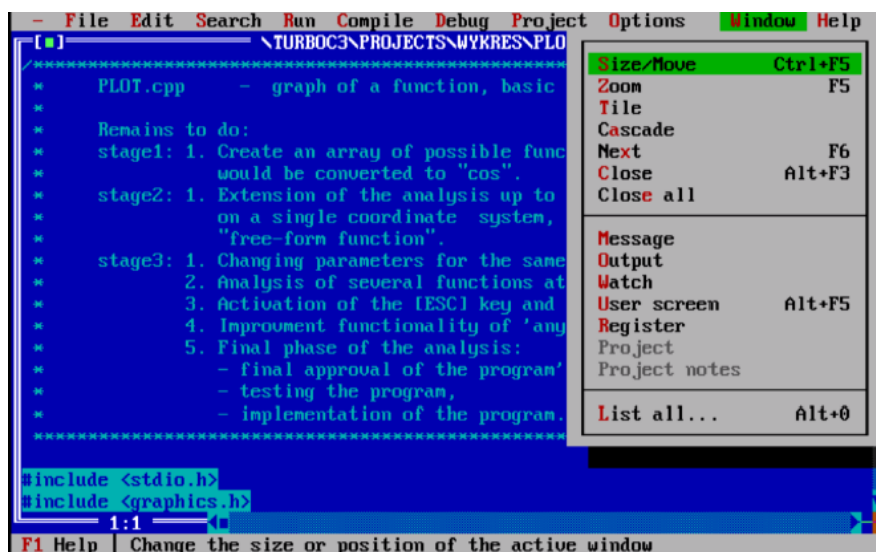
And use [Tab] and [Enter].

Here are the most important folders (and files in them):

Options --> Directories

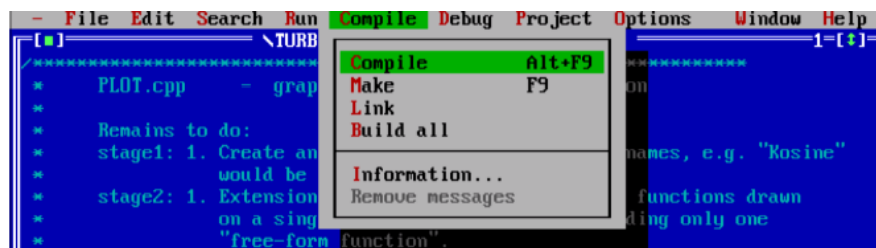


You can direct the **Output** elsewhere, but why to do that?

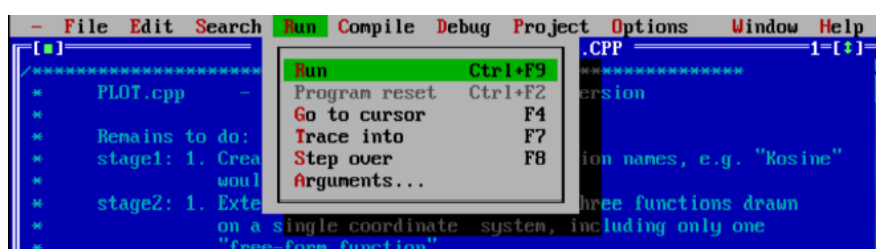


If you do not hold the displayed result with the **getch()** command, this image will disappear (almost imperceptible) and you will see the editor again. **User screen** will then give you a preview of this result.

Zoom closes **Watch**. **[F5]** closes the **Watch** and then next **[F5]** opens it again.

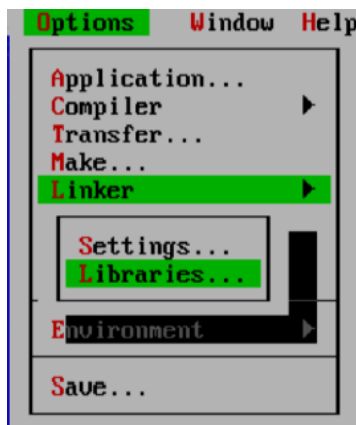


Make creates an execution file. This means that you can run such a file immediately (e.g. from your desktop), just like any file in **MSWord**. You can transfer it to another computer and run it from there - but in this case you must have **C:\TURBOC3\bgi** as declared in the program. Unfortunately, **Plot.cpp** does not work without this declaration. Maybe you can do it differently! You would have to have an old computer to do that without any problem!



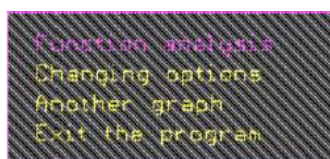
I use **Help** most often because I have everything I want there, including code examples.

To make sure that Turbo C++ has graphics connected to the program, click on **Options** -> **Linker** -> **Libraries**. Select the **Graphics Library** option and press **OK**.

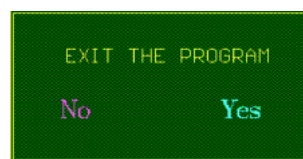


Some shortcomings of the program.

1. The mouse is not connected in software, so it does not work.
2. You cannot go far beyond the origin of the coordinate system.
3. The graph consists of thin lines and it would seem that they should be thickened. Technically it is very simple. However, this would sometimes result in visually incorrect reading of these lines. For example, $f(x)=\sin(x)/x$, here it should be visible that $X=0$ does not belong to the domain of the function. What to do if the graph slides along the lines of the coordinate system - in this case they should not be covered by the graph.
4. A line whose next point extends beyond the top or bottom of the screen and the function is still continuous is not drawn. It could be drawn to the border of the screen for the same X value as before, or $X + \text{one pixel}$. Similarly, a line could be drawn from a point off-screen to the screen without causing distortions of a function drawn.
5. Not always well entered function expression gives a plot. This shortcoming also occurs in other web applications.
6. The **outtext()** function does not accept non-English characters, although this could be manage somehow.
7. Why calling the **function_menu()** module again (second time) does not produce the desired **outtext()** result? The first call to **function_menu()** on the function graph gives the correct result:



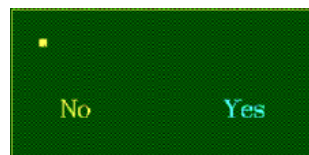
and



After entering '**Function Analysis**', checking the results, we exit the '**Function Analysis**' returning to the full picture of the function graph. However, when we want to enter the same **function_menu()** again, we are getting this:



and



function_menu() will still work and the first point is still an active '**Function Analysis**' giving the previous - as expected - data after pressing **[Enter]**.

But as you can see, calling the **function_menu()** again is not pleasing to the eye. You need to remember the order of the options:

1. Function analysis,
2. Changing options,
3. Another graph,
4. Exit the program.

There is not a logical error in the program. I found an explanation now on the Internet (in 1998, when I created this program, I had no access to the Internet):

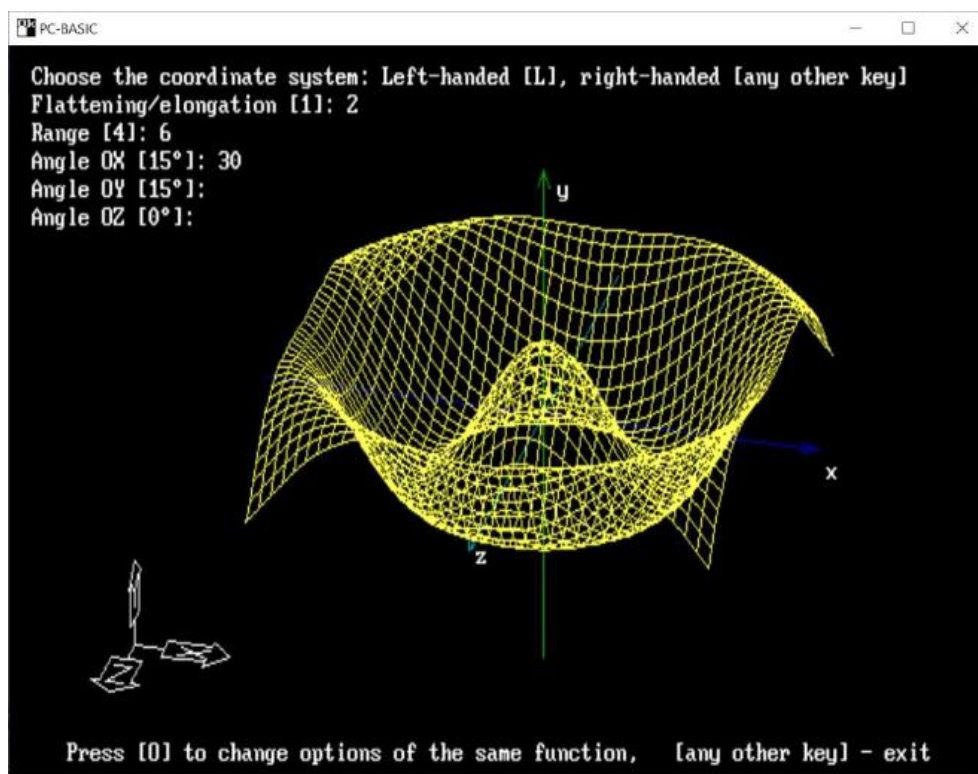
The `outtext()` function is a C function that displays text at the current position on the screen in graphics mode. However, it only works once for each element that you want to display. If you try to use it twice for the same element, it will not work because the element is already in the document and cannot be appended again. You need to create a new element for each text that you want to display, or use the `cloneNode()` method to make a copy of the existing element. Alternatively, you can use the `outtextxy()` function, which displays text at a specified point (x, y) on the screen. This function allows you to display multiple texts at different positions without creating new elements.

My intention is to share the programming logic, not to try tricks.

What about three dimensional (3D) plot?

There is a 'XYZ' program in GW-Basic below this website for a spatial figure plot, with a full code of course. It was my first PC program written over 30 years ago (1991). And here is an example of how the program works for the function `cos(sqrt(x^2+y^2))`

Instead of 'sqrt' there is 'sqr' in GW-Basic. So do not be surprised that I made 'sq' out of it in the **Plot.cpp** program!



Comparison of the names of the Polish and English versions

Program

Polish	English
Wykres.cpp	Plot.cpp

Functions

Polish	English
void main (void)	void main (void)
void wprowadzenie (void);	void introduction (void);
void wybieranie_postaci_funkcji (void);	void function_type_selection (void);
void okno_najwieksze (void);	void the_largest_window (void);
void wybieranie_wielomianu (void);	void polynomial_choosing (void);
void usun_spacje_i_analizuj (void);	void remove_spaces_and_analyze (void);
void popraw_zapis_liczby (void);	void correct_notation_of_the_number (void);
void zbadaj_wielkosc_liczby (void);	void check_number_size (void);
void wybieranie_funkcji_dowolnej (void);	void free_form_function_choosing (void);
void pytanie_o_zapis (void);	void question_about_notation (void);
void tablica_operatorow_i_funkcji (void);	void operators_and_math_func_table (void);
void grafika_wstepu (void);	void introduction_graphics (void);
void wybory_parametrow (void);	void parameters_selection (void);
void dlugosc_jednostki_osi_OX (void);	void OX_axis_unit_length (void);
void dlugosc_jednostki_osi_OY (void);	void OY_axis_unit_length (void);
void nr_wzoru_tla_ekranu (void);	void background_pattern_number (void);
void okno_wzorow_tla_i_barw (void);	void bgr_patterns_and_colors_window (void);
void nr_koloru_tla_ekranu (void);	void background_color_number (void);
void nr_koloru_osi_ukladu_wspolrz (void);	void coor_system_axes_color_number (void);
void nr_koloru_wykresu_funkcji (void);	void function_graph_color_number (void);
void wstep_rutynowy (void);	void routine_introduction (void);
void wstepna_analiza_znakow (void);	void preliminary_analysis_of_chars (void);
void brak_mozliwosci_poprawiania (void);	void no_possibility_of_correction (void);
void tlo_wykresu_funkcji (void);	void function_graph_background (void);
void wykres_funkcji (void);	void graph_of_the_function (void);
Error report	
void nie_liczba_rzeczywista (void);	void this_is_not_a_real_number (void);
void za_dlugi_lancuch (void);	void string_is_too_long (void);
void usun_spacje (void);	void remove_spaces (void);
void nie_liczba_naturalna (void);	void this_is_not_a_natural_number (void);
void liczba_poza_zakresem (void);	void out_of_range_number (void);
void kolor_ten_sam (void);	void the_same_color_is_not_allowed (void);
void wyczysc_raport_bledu (void);	void clear_error_report (void);
void wyczysc_informacje (void);	void clear_info (void);
void znaleziono_blad (void);	void error_found (void);
Exit the program and help	
void czy_porzucic_program (void);	void whether_to_exit_the_program (void);
void zdania_sprzeczne_ze_soba (int ilosc_skokow);	void contradictory_sentences (int hop_count);
void czy_porzucic_program_gr (void);	void whether_to_exit_the_program_gr (void);
void pomoc (void);	void help_non_gr (void);
void pomoc_gr (void);	void help_gr (void);
Functions and their types for an "free-form function"	
void wpisz_lancuch_i_analizuj_go (void);	void read_the_string_and_analyze_it (void);
void popraw_wyrazenie (void);	void correct_algebraic_expression (void);
void czy_jest_argument (void);	void is_there_any_argument (void);
void dzialania_w_nawiasie_wewn_test (void);	void inner_parenth_operations_test (void);
void dzialania_w_nawiasie_wewn (void);	void inner_parenth_operations (void);
void usun_tylko_nawiasy (void);	void remove_only_parentheses (void);
void wprowadz_liczbe_do_lancucha (void);	void insert_number_into_the_string (void);
float wynik_prostego_dzialania (float aa, float bb);	float result_of_a_simple_operation (float aa, float bb);
Other basic (apart from read the string and analyze it) elements of "free-form functions"	
void funkcja_dowolnej_postaci (void);	void free_form_function (void);
void oblicz_wartosc_funkcji_matem (void);	void math_function_calculation (void);
int czy_brak_obliczen (void);	int is_it_a_lack_of_calculations (void);
The "menu" window after the graph is completed	
void menu (void);	void function_menu (void);
void analiza_funkcji (void);	void function_analysis (void);
void okno_analazy_funkcji (void);	void window_of_function_analysis (void);

Variables

Polish	English	Note
int opcja = 0;	int option = 0;	Initial 0 - without meaning; Operational value from 1 to 7 to manage selection of parameters.
int kolor_tla_ekranu_opcji = 11;	int backgr_color_for_option = 11;	Screen background color for choosing parameters; initially light cyan.
int kolor_tla_wyboru_parametrow = 7;	int backgr_color_for_parameter_sel = 7;	Screen background color for parameters selection; initially light gray.
int zle;	int bad;	Flag: 0 - OK; 1 - something is wrong. Request to correct the entry.
char znak;	char character_entered;	Single key pressed for getch().
char *ptr_znak;	char *ptr_character_entered;	Pointer to 'character entered'.
int dl_lancucha;	int string_length;	Length of a string entered by the user.
int proba_poprawienia = 0;	int trying_to_correct = 0;	Flag: 0 - OK; 1 - correction needed.
int pozycja_kursora_x;	int cursor_position_x;	Horizontal position of a corsor for a non-graphic screen; taking from wherex() build-in function.
Int pozycja_kursora_y;	Int cursor_position_y;	Vertical position of a corsor for a non-graphic screen; taking from wherey() build-in function.
int sygnal = 0;	int beep_attention = 0;	All purpose flag for a specific use. Request to correct the entry.
Choosing parameters		
int dl_jedn_OX = 50;	int unit_length_on_the_OX = 50;	Operational value of 'Unit length on OX axis in pixels' for final plot; initial 50; acceptable range [10;300].
int dl_jedn_na_osi_OX = 50;	int unit_length_on_the_OX_axis = 50;	The same as above, but while drawing a function. Here: fixed value remains unchanged.
int dl_jedn_OY = 50;	int unit_length_on_the_OY = 50;	Operational value of 'Unit length on OY axis in pixels' for the final plot; initial 50; acceptable range [5;220].
int dl_jedn_na_osi_OY = 50;	int unit_length_on_the_OY_axis = 50;	The same as above, but while drawing a function. Here: fixed value remains unchanged.
int zmniejszenie_jednostki_OY = 1;	int unit_OY_length_reduction = 1;	1 - do not change 'unit length on OY axis in pixels' you entered for the final plot; Other - decrease the 'unit length on OY axis in pixels' you entered 10 times.
int nr_wz_tla = 1;	int backgr_patt_number = 1;	Operational value of 'Background pattern number' for the final plot; 1 - solid color; acceptable range [0;11].
int nr_wzoru_tla = 1;	int backgr_pattern_number = 1;	The same as above, but while drawing a function. Here: fixed value remains unchanged.
int nr_kol_tla = 7;	int backgr_col_number = 7;	Operational value of 'Background color number' for the final plot; 7 - light gray; acceptable range [0;15].
int nr_koloru_tla = 7;	int backgr_color_number = 7;	The same as above, but while drawing a function. Here: fixed value remains unchanged.
int nr_kol_osi = 12;	int axes_col_number = 12;	Operational value of 'Axes of the coordinate system color number' for the final plot; 12 - light red; acceptable range [0;15].
int nr_koloru_osi = 12;	int axes_color_number = 12;	The same as above, but while drawing a function. Here: fixed value remains unchanged.
int nr_kol_wykresu = 10;	int function_plot_col_number = 10;	Operational value of 'Function graph color number' for the final plot; 10 - light green; acceptable range [0;15].
int nr_koloru_wykresu = 10;	int function_plot_color_number = 10;	The same as above, but while drawing a function. Here: fixed value remains unchanged.
For 'polynomial function' choosing only		
int ilosc_kropek_dziesietnych;	int number_of_decimal_points;	For 'Polynomial choosing' only
int wielomian = 0;	int polynomial_chosen = 0;	Flag: 0 - polynomial not chosen, 1 - polynomial was chosen - opposite of 'free_form_function_chosen'.
For 'free-form function' choosing only		
int funkcja_dowolna = 0;	int free_form_function_chosen = 0;	Flag: 0 - free-form function not chosen, 1 - free-form function was chosen - opposite of 'polynomial_chosen'.
int jest_cyfra = 0;	int it_is_a_digit = 0;	Flag: 0 - it is not a digit, 1 - digit is detected.
int jest_x = 0;	int x_found = 0;	Flag: 0 - variable (x) was not found, 1 - variable (x) was found.
int zmniejsz_d_l;	int decrease_d_l;	Counter: How many places in the entered

		string reduce its length. E.g.: By removing spaces.
int wskaznik;	int indicator;	All purpose indicator for entered string - detects 'x', digit, ...
int dodawanie = 0;	int addition = 0;	Flag: 0 - it is not an addition, 1 - it is an addition.
int odejmowanie = 0;	int subtraction = 0;	Flag: 0 - it is not a subtraction, 1 - it is a subtraction.
int mnozenie = 0;	int multiplication = 0;	Flag: 0 - it is not a multiplication, 1 - it is a multiplication.
int dzielenie = 0;	int division = 0;	Flag: 0 - it is not a division, 1 - it is a division.
int dzialanie;	int is_there_any_operator_indic;	Flag: 0 - this is not like +, -, *, / 1 - one of +, -, * and / was detected.
int nr_nawiasu_otwartego;	int number_of_bracket_open;	The next number of the open parenthesis.
int nr_nawiasu_zamknietego;	int number_of_bracket_closed;	The next number of the closed parenthesis.
int poczatek_operacji;	int operation_begins;	Position in a string where calculation begins, e.g. the innermost parentheses.
int koniec_operacji;	int operation_ends;	Position in a string where calculation ends, e.g. the innermost parentheses.
int pozycja_nawiasu;	int parenthesis_position;	Position in a string where one of parentheses, '(' or ')' is inserted.
int jest_funkcja;	int it_is_a_function;	A math function detector. Flag: 0 - so far there is not a function, 1 - here a math function begins.
int jest_?;	int it_is_?;	where: '?' could be ln, lg, cos, sin, tan, cosh, sinh, tanh, acos, asin, atan, abs, mod, pow, exp, sq.
int poz_?_1;	int pos_?_1;	
int poz_?_2;	int pos_?_2;	
int operacje_na_funkcji = 0;	int operations_on_function = 0;	void math_function_calculation(void) Flag: 0 - do not treat it as a math function, 1 - start treating that as a math function and finally run math_function_calculation() module.
int ilosc_przecinkow_faktyczna;	int actual_number_of_commas;	Number of commas inside mod() or pow() function on a entered string.
int ilosc_przecinkow_przewidywana;	int predicted_number_of_commas;	Expected number of commas inside mod() or pow() function (=1).
Int jest_przecinek = 0;	int it_is_comma = 0;	„Commas” indicator. Flag: 0 - no comma, 1 - it is a comma, 2 - ready to take the second argument for either mod() or pow() function.
int porzuc;	int terminate;	Flag: 0 - still draw the function 1 - stop drawing the line because of a specific issue, e.g. next point is out of the screen or the next point does not belong to the function domain.
int jest_problem_z_dziedzina = 0;	int problem_with_domain = 0;	Flag: 0 - no problem with a function domain, 1 - problem with a domain, e.g. An exponentiation result is too high (this program limitation).
int blad = 0;	int error_detector = 0;	All purpose flag for error detecting. If any, the 'error_found()' function is called.
int dziedzina;	int function_domain;	Flag: 0 - the point 'x' does not belong to the function domain, 1 - this point 'x' belongs to the function domain.
int kwadrat_maksymalny[];	int maximum_square[];	void introduction(void) void introduction_graphics(void) void function_graph_background(void) Set of variables that start a graphic screen.
unsigned rozmiar;	unsigned image_size;	
void far *ptr_do_alokacji_pamieci;	void far *ptr_to_memory_allocation;	void remove_spaces_and_analyze(void) Flag: 0 - this is not a digit, 1 - this is a digit.
int jest_cyfra = 0;	int it_is_a_digit = 0;	
int jest_cyfra_znaczaca = 0;	int it_is_a_significant_digit = 0;	void correct_notation_of_the_number(void) Avoid printing insignificant digits and remove excess of dots.
int usun_tylko_kropke = 0;	int delete_only_the_dot = 0;	
int d_l_fikcja;	int d_l_fiction;	void read_the_string_and_analyze_it(void) Takes 'Length of the entered string' for a special purposes, ex. entering 3.141593 instead of 'pi' (π) or 2.718282 instead of 'e'.
'Help' - for both graphic and non-graphic screens		
int wezwanie_pomocy = 0;	int calling_for_help = 0;	Flag: 0 - no 'help' window needed, 1 - 'help' window needed. For both graphic and non-graphic screen.
int strona;	int page_number;	Page number for the 'help_non_gr()' function.

char another_character;	char znak_dodatkowy;	void help_non_gr(void) For [PgDn] and [PgUp] using getch() in 'help_non_gr()' function.
petla_?;	page_no_?;	void help_non_gr(void) labels Where ? means: 1, 2, 3, 4, 5 or 6 To manage sequence of six windows the same size for the 'help_non_gr()' function.
rozdzielnia;	switchboard;	void help_non_gr(void) label To allow works of set of 'page_no_?'.
int pole_pomocy_gr[];	int help_field_gr_box[];	void help_gr(void) Window size definition for the function.
Exit the program		
int proba_porzucenia = 0;	int trying_to_exit = 0;	Flag: 0 - ready to not exit the program, 1 - ready to exit the program. For both graphic and non-graphic screens.
int pole_porzucenia_programu[];	int exit_the_program_box[];	void whether_to_exit_the_program_gr(void) Window size definition for the function.
int ilosc_skokow = 0;	int hop_count = 0;	Counter of switching between sentences for 'contradictory_sentences(hop_count)' function.
void graph_of_the_function(void)		
float granica_rozsadku;	float limit_of_reason;	Arbitrarily (according to my own logic) imposed upper (and automatically lower as 'minus upper') quantitative limits on a number of variables with values in natural numbers and, in addition, other numerical limits for the same variable but treated differently. For example, the 'unit length of the OX axis' < 20 is 1e+18, and for > 20 and larger, the 'limits of reason' are smaller and smaller numbers.
int yy_poprzedni;	int yy_previous;	Saved position on the screen of the previous point for possible reference. Then the variable point_flag = 1 and optionally draws a line. Offscreen: Does not draw a line.
int punkt_poprzedni;	int point_previous;	Flag: 0 - there is no such point, 1 - there is a point on the screen. Is there any previous point that can be referred to? E.g., when drawing is interrupted - discontinuity, going off-screen, etc. - there is no such point.
int punkt_poprzedni_poprzedni;	int point_before_previous;	Flag: 0 - there is no such point 1 - there is a point on the screen Is there any point before the previous point that can be referred to?
float y_poprzedni;	float y_previous;	Actual previous point value of 'Y'.
float y_poprzedni_poprzedni;	float y_before_previous;	Actual before previous points value of 'Y'.
float skladnik;	float summand;	The numeric summand.
float wartosc_piksela_OX;	float OX_pixel_value	Find the X coordinate of a pixel on the screen.
float dl_jedn_na_osi_OY_float;	float unit_length_on_the_OY_axis_flo;	Find the Y coordinate of a pixel on the screen. The variable it to avoid rounding. (Pointing to an exact pixel could cause not a smooth line.)
int jest_punkt = 0;	int point_flag;	Point position has been calculated and ... Flag: 0 - ... point is off screen, 1 - ... point is on the screen. For detecting current point on the screen.
Some other variables		
int dziedzina_poprzednia;	int previous_function_domain;	An auxiliary flag for estimating the oundaries of the domain of free-form function.
int brak_obliczen = 0;	int lack_of_calculations = 0;	int is_it_a_lack_of_calculations(void) Flag: 0 - go ahead and do calculation, 1 - do not do calculation. Do not calculate if some condition is met.
int pole_menu[];	int menu_box[];	void function_menu(void) Window size definition for the function.
int pole_analizy_funkcji[];	int function_analysis_box[];	void window_of_function_analysis(void) Window size definition for the function.
int pole_bledu[];	int error_box[];	void error_found(void) Window size definition for the function.

The characters used in the program that are available on the English code page (437) and at the same time unavailable on the Polish code page (852), and therefore probably also unavailable on another non-English code page:

Character	Code page 852 number
≈	247
α	224
π	227
ε	238 (it replaces €)

Typing numbers with the decimal comma

Entering numbers with a decimal comma instead of a decimal point can be easy: Don't allow typing a "period" and enter a line of code that replaces the visual "comma" with the "dot" to enable calculations on the entered number.

At the same time each character is verified. This program is for non-English users.

It is true that a number can be verified after entering it, but control over each character allows for a better course of this process.

This program can be used as a function that is called whenever you need to type a number (it can be both, either an integer or a floating point number) from the keyboard.

If we need to print the final (after calculations in the program) number with a **decimal comma** (not with a decimal point), we would have to:

- replace a number with a string (in 'C' it is a function of itoa(), that is mean **integer to alphabetic**),
- find the position of a dot (if any),
- replace the dot with comma,
- print this string (pretending it is a number).

Code example:

Turbo C++

We do not need to use the 'setlocale()' function, but it does not create any problem. However the program is prepared also for Dev C++ environment when we exchange one line to clean the output screen: From **clrscr();** to **system("cls");**

```
/* *****
 * File: Number.C
 * *****
 * Entering a number from the keyboard and the same time verifying each character.
 * This program is for non-English users because a decimal comma (e.g. 12,34) is
 * used instead of decimal dot (e.g. 12.34).
 */

#include<stdio.h>
#include<conio.h>      // for getch()
#include<ctype.h>      // for isdigit()
#include<locale.h>     // for non-English national characters (eg. German)
#include<stdlib.h>     // for system("cls")

float a;              // final result of entered number
char A[15];           // collecting digits to convert their string into a number
int flag = 0;         // flag=0 <-- no error; flag=1 <-- an error has occurred
int i=0, j, k, l;     // 'i' is only to avoid the first (i=0) number entered to
                     // be equal to zero, e.g. in the expression of
                     // a*x**2 + b*x (for the quadratic equation)
                     // 'j' - the next digit in the number
                     // 'k' - variable needed to clear an error information
                     // 'l' - comma counter in the number

void main()
```

```

{
    setlocale(LC_CTYPE, "Polish");
    // setlocale( LC_ALL, "German");    // This is an example to set the German
                                        // language

    clrscr();                          // Clear the screen
    // system("cls");                  // for 'Dev C++'

    printf("\n Entering a number from the keyboard and the same time verifying each
character");
    printf("\n                In the entering number can be a decimal comma (not a dot)");
    printf("\n                The [ENTER] takes the number. Do not write in English!
\n");
    printf("\n                After \'CORRECT IT\' press any key for correction
\n\n");
    printf("\n Enter a number here: ");

    j = 0;                            // next element of the array
    l = 0;                            // number of commas
    while( ( A[j] = getch() ) != '\x0D') // type characters. When you are done, press
                                        // [ENTER]
    { flag = 0;
    // if(i == 0)                      // only if the number cannot be zero
    // if(j == 0 && A[j]=='0')
    // { printf(" 'a' cannot be 0      CORRECT IT");
    //   flag = 1;
    // }

    if(j == 0)                        // for any number
    { if( !(A[j] == '-' || A[j] == ',' || isdigit(A[j])) )
      { printf(" Acceptable are -, digit. CORRECT IT");
        flag = 1;
      }
    }
    else
    { if( A[j] == ',')                // replace the comma with the dot for further
                                      // calculations on that number
      { A[j] = '.';
        l++;
      }
    }
    else
    { if(A[j] == '.')
      { printf(" Do not write in English CORRECT IT");
        flag = 1;
      }
    }
    else
    { if( A[j] == ',')                // replace the comma with the dot for further
                                      // calculations on that number
      { A[j] = '.';
        l++;
        if( l > 1 )
        { printf(" Too many commas      CORRECT IT");
          flag = 1;
        }
      }
    }
    else
    { if( !isdigit(A[j]) )
      { printf(" Must be a digit here ! CORRECT IT");
        flag = 1;
      }
    }

    if( flag == 1 )
    { getch();                        // hold the error message

    for(k=0;k<40;k++) printf("\b"); // move the cursor 39 positions back
    for(k=0;k<40;k++) printf(" "); // type 39 spaces to erase the information
    for(k=0;k<40;k++) printf("\b"); // move the cursor 39 positions back

```



```

    }
    else
    {
        j++;          // character was accepted; preparing to enter the next one
    }                // end of the 'while' loop

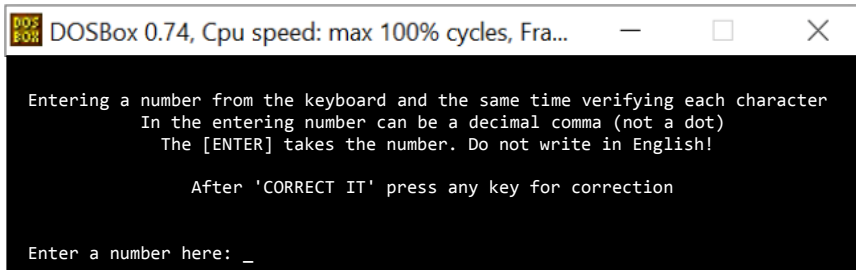
    A[j++] = '\0';    // close the string
    a = atof(A);      // exchange the string (A) into the number (a)
    printf("\n\n The program has accepted the number %f for further calculations on it\n\n", a);

    printf("\n\n                               Press any key to exit ");
    getch();

}                    // end of the 'main' function

```

Possible results:



DOSBox 0.74, Cpu speed: max 100% cycles, Fra...

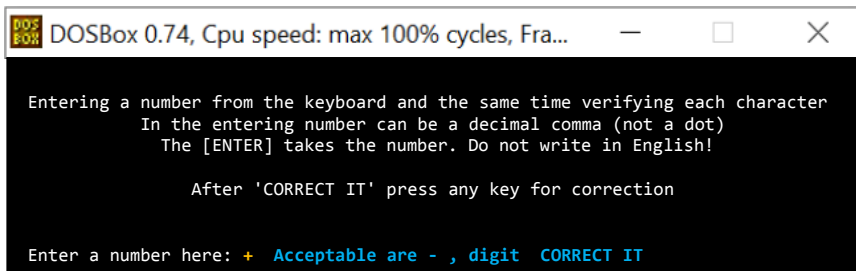
```

Entering a number from the keyboard and the same time verifying each character
In the entering number can be a decimal comma (not a dot)
The [ENTER] takes the number. Do not write in English!

After 'CORRECT IT' press any key for correction

Enter a number here: _

```



DOSBox 0.74, Cpu speed: max 100% cycles, Fra...

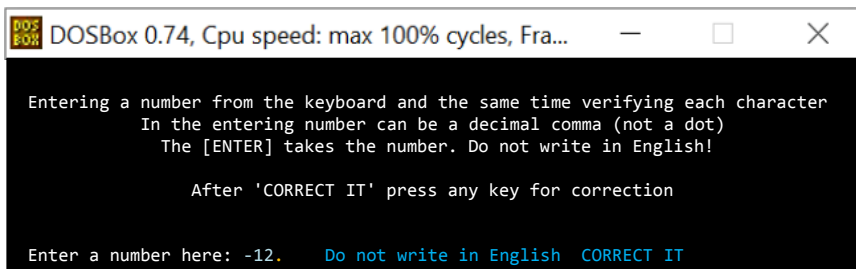
```

Entering a number from the keyboard and the same time verifying each character
In the entering number can be a decimal comma (not a dot)
The [ENTER] takes the number. Do not write in English!

After 'CORRECT IT' press any key for correction

Enter a number here: + Acceptable are - , digit CORRECT IT

```



DOSBox 0.74, Cpu speed: max 100% cycles, Fra...

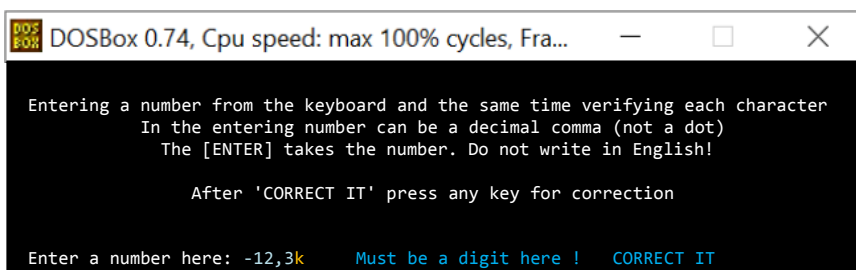
```

Entering a number from the keyboard and the same time verifying each character
In the entering number can be a decimal comma (not a dot)
The [ENTER] takes the number. Do not write in English!

After 'CORRECT IT' press any key for correction

Enter a number here: -12. Do not write in English CORRECT IT

```



DOSBox 0.74, Cpu speed: max 100% cycles, Fra...

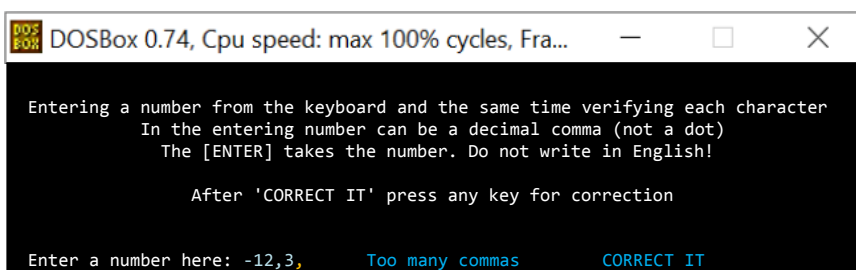
```

Entering a number from the keyboard and the same time verifying each character
In the entering number can be a decimal comma (not a dot)
The [ENTER] takes the number. Do not write in English!

After 'CORRECT IT' press any key for correction

Enter a number here: -12,3k Must be a digit here ! CORRECT IT

```



DOSBox 0.74, Cpu speed: max 100% cycles, Fra...

```

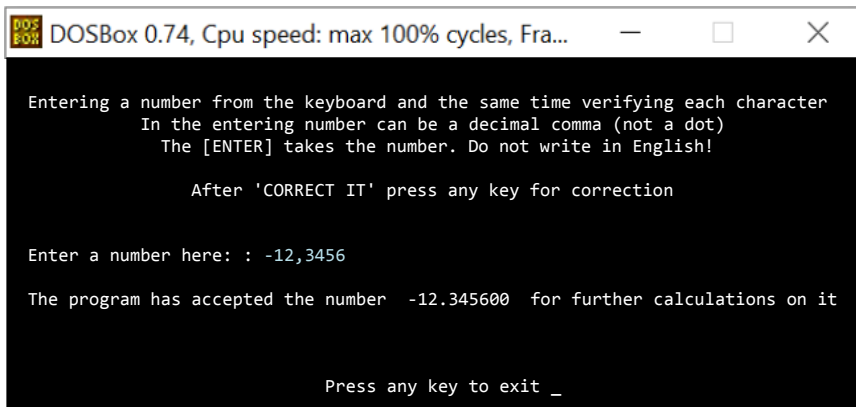
Entering a number from the keyboard and the same time verifying each character
In the entering number can be a decimal comma (not a dot)
The [ENTER] takes the number. Do not write in English!

After 'CORRECT IT' press any key for correction

Enter a number here: -12,3, Too many commas CORRECT IT

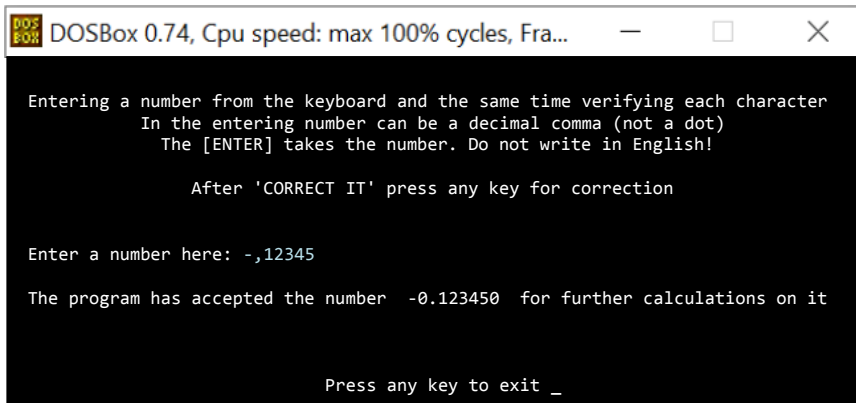
```

finally:



```
DOSBox 0.74, Cpu speed: max 100% cycles, Fra...  
  
Entering a number from the keyboard and the same time verifying each character  
In the entering number can be a decimal comma (not a dot)  
The [ENTER] takes the number. Do not write in English!  
  
After 'CORRECT IT' press any key for correction  
  
Enter a number here: : -12,3456  
  
The program has accepted the number -12.345600 for further calculations on it  
  
Press any key to exit _
```

Another example:



```
DOSBox 0.74, Cpu speed: max 100% cycles, Fra...  
  
Entering a number from the keyboard and the same time verifying each character  
In the entering number can be a decimal comma (not a dot)  
The [ENTER] takes the number. Do not write in English!  
  
After 'CORRECT IT' press any key for correction  
  
Enter a number here: -,12345  
  
The program has accepted the number -0.123450 for further calculations on it  
  
Press any key to exit _
```