Міністерство науки і освіти, молоді та спорту України

Національний технічний університет України

“Київський політехнічний інститут”

Основи програмування та алгоритмічні мови – 2.

Спеціальні засоби мови програмування

ЗВІТ ДО

КОМП’ЮТЕРНОГО ПРАКТИКУМУ No 10

«Наукова графіка»

(ТЕМА)

Варіант No 3

Дата «\_\_\_» \_\_\_\_\_\_\_ 20\_\_\_

Оцінка «\_\_\_\_\_\_\_\_\_»

Дата «\_\_\_» \_\_\_\_\_\_\_ 20\_\_\_

Виконав: студент \_\_\_ курсу

гр. ЗПІ-ЗП73

Боярко Оксана Олександрівна

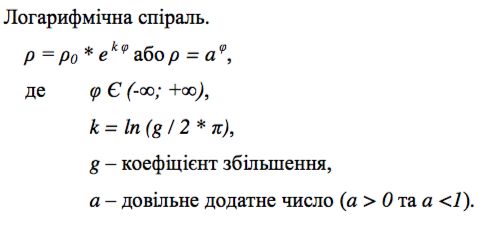
*(П.І.Б.)*

Перевірив: Крячок О.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(П.І.Б., підпис)*

Київ 2018

* Завдання до роботи
* необхідно написати програму, що виконує розрахунок за певним рівнянням та будує графік функції.
* 
* Додатково необхідно у правому верхньому кутку екрана відобразити «числовий» годинник, у лівому верхньому кутку надати текстову інформацію про автора програми, а в нижній частині екрана вивести рядок з підказкою про активні клавіші для управління роботою програми.
* Теоретичні відомості.

Для побудови графіку у полярних координатах потрібно використати формулу перетворення у декартову систему координат.

ρ = a \* sin (5 φ) або ρ = a \* cos (5 φ)

Для відображення графіку можна використовувати SDL2 бібліотеку. Це кросплатформена мультимедійна бібліотека, що реалізує єдиний програмний інтерфейс до графічної підсистеми.

SDL\_RenderDrawLine – рисує лінію з заданими координатами

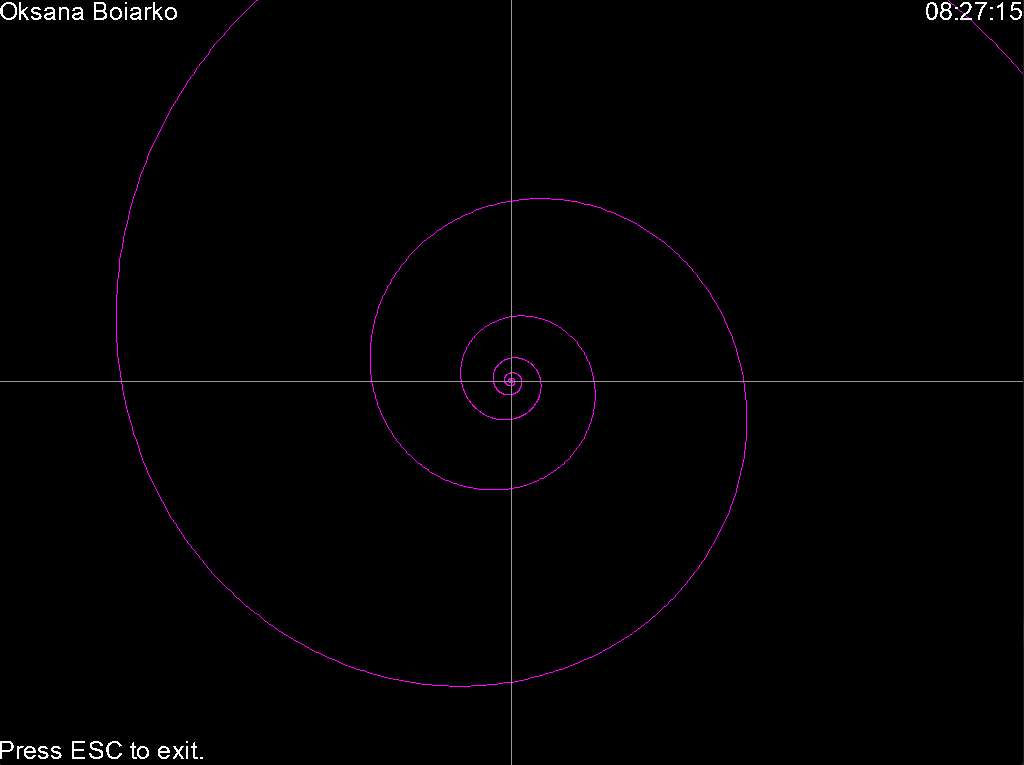
SDL\_PollEvent(&event) – функція задає, на які івенти буде реагуати вікно з графіком

SDL\_DestroyRenderer – закриває процес рендеру графіку

Так як бібліотека не має дефолтних даних про шрифт, то потрібно импортувати його у форматі ttf.

TTF\_RenderText\_Solid – функція рендеру тексту

Блок-схема

* Результати роботи.
* 

Висновки по роботі.

У роботі викристовувались бібліотеки SDL.h та SDL\_ttf.h. Переведення полярних координат у декартові для відрисунку було проведено за допомогою стандартної бібліотеки математичних формул math.h

* Програмний код (Додаток).
* #include <string.h>  
  #include <SDL2/SDL.h>  
  #include <SDL2\_ttf/SDL\_ttf.h>  
  #include <iostream>  
  #include <cmath>  
  #include <ctime>  
    
  **using namespace** std;  
    
  **void** render\_text(SDL\_Renderer \*renderer, **int** x, **int** y, **const char** \*text, TTF\_Font \*font, SDL\_Rect \*rect, SDL\_Color \*color);  
    
  **int** main(**int** argc, **char**\* argv[]) {  
   **double** g = 0.75;  
   **double** k = log (g / 2 \* M\_PI);  
   **double** a = 0.001;  
   **int** MAX = 360;  
    
   **double** width = 1024;  
   **double** height = 768;  
   **double** x\_offset = width/2;  
   **double** y\_offset = height/2;  
    
   time\_t raw\_time;  
   **struct** tm \*timeinfo;  
   **char** buffer[80];  
    
   **if** (SDL\_Init(SDL\_INIT\_EVERYTHING) == 0) {  
   SDL\_Window \*window = NULL;  
   SDL\_Renderer \*renderer = NULL;  
   TTF\_Init();  
   window = SDL\_CreateWindow( "Lab 10", 0, 0, width, height, 0);  
   renderer = SDL\_CreateRenderer( window, -1, *SDL\_RENDERER\_ACCELERATED*);  
    
   SDL\_Color White = {255, 255, 255, 255};  
   TTF\_Font \*font = TTF\_OpenFont("arial.ttf", 25);  
    
   **int** w, h;  
   TTF\_SizeText(font, "99:99:99", &w, &h);  
   SDL\_Rect time\_rect;  
    
   **bool** whirl = **true**;  
   SDL\_SetRenderDrawColor( renderer, 0, 0, 0, 255);  
    
   **if** (window != **nullptr** && renderer != **nullptr**) {  
   SDL\_bool done = *SDL\_FALSE*;  
   **while** (!done) {  
   SDL\_Event event;  
   time(&raw\_time);  
   timeinfo = localtime(&raw\_time);  
   strftime (buffer,80,"%I:%M:%S",timeinfo);  
    
   SDL\_SetRenderDrawColor(renderer, 0, 0, 0, SDL\_ALPHA\_OPAQUE);  
   SDL\_RenderClear(renderer);  
    
   SDL\_SetRenderDrawColor(renderer, 255, 0, 255, SDL\_ALPHA\_OPAQUE);  
    
   **double** dX\_p = (a \* pow(M\_E, k\*0)) \* cos(0) + x\_offset;  
   **double** dY\_p = (a \* pow(M\_E, k\*0)) \* sin(0) + y\_offset;  
    
   **for** (**int** i = 1; i < MAX\*100; i++) {  
   **double** theta = i \* 2 \* M\_PI / MAX;  
   **double** dX = (a \* pow(M\_E, k\*theta)) \* cos(theta) + x\_offset;  
   **double** dY = (a \* pow(M\_E, k\*theta)) \* sin(theta) + y\_offset;  
    
   SDL\_RenderDrawLine(renderer, dX\_p, dY\_p, dX, dY);  
   dX\_p = dX;  
   dY\_p = dY;  
   }  
    
   // SPINNER  
   **if** (k > 0.1 && whirl) {  
   k \*= 0.999;  
   }  
   **else** {  
   k \*= 1.001;  
   whirl = **false**;  
   }  
    
   **if** (k > 0.5)  
   whirl = **true**;  
    
   SDL\_SetRenderDrawColor(renderer, 150, 150, 150, 30);  
   SDL\_RenderDrawLine(renderer, 0, y\_offset, width, y\_offset);  
   SDL\_RenderDrawLine(renderer, x\_offset, 0, x\_offset, height);  
   SDL\_SetRenderDrawColor(renderer, 255, 255, 255, SDL\_ALPHA\_OPAQUE);  
   render\_text(renderer, 0, 0, "Oksana Boiarko", font, &time\_rect, &White);  
   render\_text(renderer, width - w, 0, buffer, font, &time\_rect, &White);  
   render\_text(renderer, 0, height - h, "Press ESC to exit.", font, &time\_rect, &White);  
   SDL\_RenderPresent(renderer);  
    
    
   **while** (SDL\_PollEvent(&event)) {  
   **if** (event.type == *SDL\_QUIT* || event.key.keysym.scancode == *SDL\_SCANCODE\_ESCAPE*) {  
   done = *SDL\_TRUE*;  
   }  
   }  
   }  
   }  
    
   **if** (renderer) {  
   SDL\_DestroyRenderer(renderer);  
   }  
    
   **if** (window) {  
   SDL\_DestroyWindow(window);  
   }  
    
   TTF\_CloseFont(font);  
   TTF\_Quit();  
   }  
    
   SDL\_Quit();  
   **return** 0;  
  }  
    
    
  **void** render\_text(SDL\_Renderer \*renderer,  
   **int** x,  
   **int** y,  
   **const char** \*text,  
   TTF\_Font \*font,  
   SDL\_Rect \*rect,  
   SDL\_Color \*color) {  
   SDL\_Surface \*surface;  
   SDL\_Texture \*texture;  
    
   surface = TTF\_RenderText\_Solid(font, text, \*color);  
   texture = SDL\_CreateTextureFromSurface(renderer, surface);  
   rect->x = x;  
   rect->y = y;  
   rect->w = surface->w;  
   rect->h = surface->h;  
    
   SDL\_FreeSurface(surface);  
   SDL\_RenderCopy(renderer, texture, NULL, rect);  
   SDL\_DestroyTexture(texture);  
  }