МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КІБЕРБЕЗПЕКИ

Курсова робота

З дисципліни:

«Технології безпечного програмування»

Виконав (ла) : Борщ Д.О., група КБ-01

Перевірив: Братушка С.М.

СУМИ – 2021

**ЗМІСТ**

[**ВСТУП**](#_14bfjjogp4hi) **3**

[**ТЕОРЕТИЧНА ЧАСТИНА**](#_n2j0bvar8o4z) **4**

[1.1 Постановка задачі](#_wepgy995ocaj) 4

[1.2 Опис об'єктно-орієнтованого підходу](#_tg2x9ozgiuhk) 4

[1.3 Опис предметної області](#_w1e2x5i5ssbc) 5

[**ПРАКТИЧНА ЧАСТИНА**](#_no8tjepxm1b7) **6**

[2.1 Опис об’єктів системи](#_imlvzglc00i1) 6

[2.2 Результати тестування програмного продукту.](#_q0428ablqd0t) 9

[**ВИСНОВКИ**](#_4zkh485z25el) **10**

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**](#_hunw2fqf5j12) **11**

[**ДОДАТКИ**](#_8cleu5qeqvg6) **12**

[Додаток А](#_k6orybnrrhzc) 12

[Додаток Б](#_sc1sxcf70f6r) 13

[Додаток В](#_r516lnrtz868) 19

# ВСТУП

Об’єктно-орієнтоване програмування[[1]](#_hunw2fqf5j12) як парадигма[[2]](#_hunw2fqf5j12) з’явилась ще в далеких 60-х роках минулого сторіччя. Але досі являється однією з основних. Більшість мов програмування або містять реалізації ООП, або є цілком об’єктно-орієнтованими.

Основною з особливостей мови С++[[3]](#_hunw2fqf5j12) э саме об’єктно-орієнтовані можливості, в порівнянні з мовою С. В ній з’являються класи, які реалізують три основні особливості ООП: інкапсуляція, успадкування, поліморфізм. Протягом курсу Технології безпечного програмування ми вивчали Об’єктно орієнтоване програмування в мові С++. І дана курсова робота повинна бути підсумуванням навичок, що ми здобули протягом цього курсу.

# ТЕОРЕТИЧНА ЧАСТИНА

## 1.1 Постановка задачі

Розробка та реалізація та опис ієрархії класів, які реалізують графічні примітиви для площини (на мові програмування Visual C++).

Графічні примітиви – лінії; прямокутники; кола, еліпси, дуги, сегменти та сектори.

## 1.2 Опис об'єктно-орієнтованого підходу

Для більш наочного виконання завдання мною було вирішено використовувати бібліотеку SDL2[[4]](#_hunw2fqf5j12) для простої двовимірної графіки, що передбачає створення

SDL(Simple DirectMedia Layer) — це кросплатформна бібліотека розробки, призначена для забезпечення низькорівневого доступу до аудіо, клавіатури, миші, джойстика та графічного обладнання через OpenGL/Direct3D/Metal/Vulkan. Він використовується програмним забезпеченням для відтворення відео, емуляторами та популярними іграми, включаючи відзначений нагородами каталог Valve та багато інших проектів. Але ми будемо використовувати його для простого попіксельного малювання в вікні, яке інтерпретуємо як координатну площину.

Також для основних графічних примітивів створимо свою ієрархію класів.

## 1.3 Опис предметної області

Перед початком роботи з графікою бібліотека SDL передбачає створення об’єкту вікна та рендереру. Тому було вирішено створити базовий клас Window, що містить покажчики на ці базові об’єкти та містить конструктор та деструктор для створення та знищення цих об’єктів та покажчиків.

Дочірні класи містять додаткові поля, для зберігання параметрів фігур та свій метод draw() для малювання фігури у вікні. Всього було створено 5 дочірніх класів:

* Line(проста лінія заданої довжини);
* Circle(коло заданого радіусу);
* Square(квадрат із заданою стороною);
* Rhombus(ромб із заданою стороною);
* Segment(сегмент кола, для його побудови використовується заданий радіус та кут ɑ).

Повна ієрархія класів виглядає наступним чином.

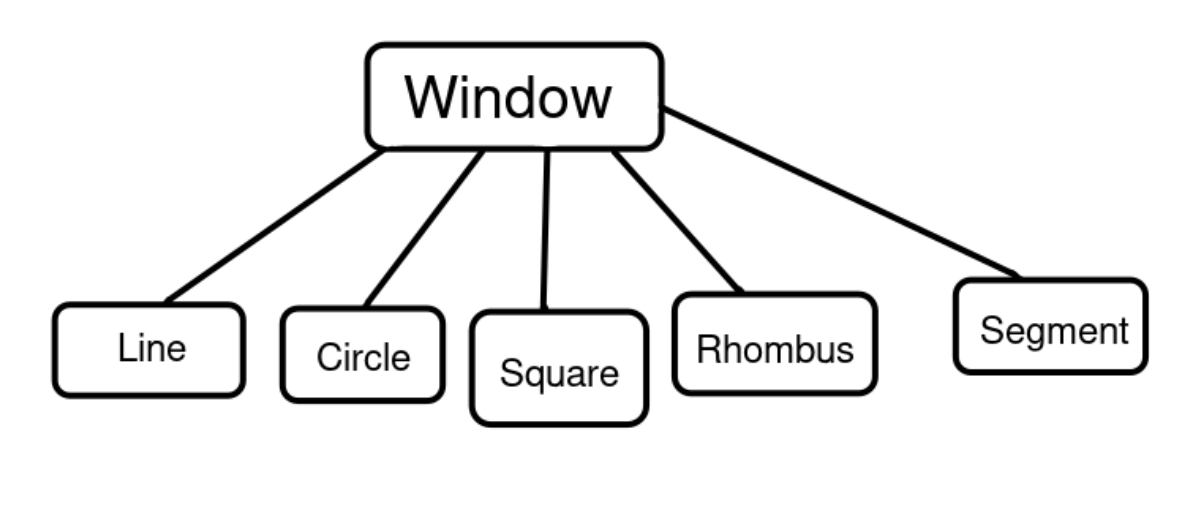


Рис 1.1 - Ієрархія класів.

# ПРАКТИЧНА ЧАСТИНА

## 2.1 Опис об’єктів системи

Робота програми починається зі створення об’єкту SDL\_Event, що може зберігати об’єкт івенту бібліотеки SDL. Далі програма ініціалізує відео в бібліотеці SDL і створює об’єкти користувацьких класів, які в свою чергу викликають конструктори, що створюють свої вікна та рендерери. Кожне вікно заповнюється чорним кольором і в центрі ставиться білий піксель, що позначає центр області побудови.

Після того, як всі об’єкти були створені та ініціалізовані, запускається main loop програми. Такий собі безкінечний цикл, який завершується лише в необхідний момент, завершуючи програму. В цьому циклі відбувається poll того самого івенту, створеного на початку. І якщо ми викликаємо завершення програми з консолі, або натискаємо на кнопку закрити у вікні, то завершуємо цикл, завершуючи програму. Деструктор головного класу Window знищує об’єкти та покажчики window та renderer.

Далі розглянемо будову кожного класу.

1. **Line**

Містить поле length, що означає довжину лінії.

Класс Line в своєму конструкторі містить визначення довжини лінії та виклик виклик метода, що змінює назву вікна, та метода, який змінює колір пікселів на зелений.

Метод *draw()* для класу Line за допомогою звичайного циклу будує

лінію заданої в полі класу довжини.

*void setLength(float)* - метод, що перевизначає довжину лінії. В якості аргумента приймає одне число float;

*float getLength()* - метод, що повертає довжину лінії.

1. **Circle**

Містить поле radius, що означає радіус кола та використовується для його побудови.

Класс Circle в своєму конструкторі містить визначення радіусу кола та виклик метода, що змінює назву вікна, та метода, який змінює колір пікселів на синій.

Метод *draw()* для класу Circle використовуючи рівняння кола та цикл будує його за двома осями, бо через невелике розширення зображення та особливості використання циклу, коло виходить неповне, тому ми добудовуємо його, проходом по іншій осі.

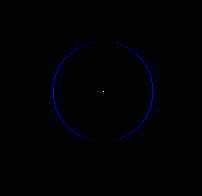


Рис 2.1 - Коло, побудоване циклом, що пройшов по одній осі.

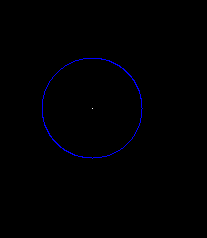


Рис 2.2 - Добудоване коло.

*void setRadius(float)* - метод, що визначає радіус кола. В якості аргумента приймає одне число float;

*float getRadius()* - метод, що повертає радіус кола.

1. **Square**

Містить полу side, що визначає

Класс Square в своєму конструкторі містить визначення сторони квадрату та виклик метода, що змінює назву вікна, та метода, який змінює колір пікселів на червоний.

Метод *draw()* для класу Square подібно до кола будує квадрат за допомогою двох циклів, перший будує лінії в горизонтальній площині, а другий у вертикальній.

*void setSide(float)* - метод, що визначає сторону квадрата. В якості аргумента приймає одне число float;

*float getSide()* - метод, що повертає сторону квадрата.

1. **Rhombus**

Класс Rhombus в своєму конструкторі містить визначення сторони ромба та виклик метода, що змінює назву вікна, та метода, який змінює колір пікселів на червоний.

В методі *draw()* для класу Rhombus подібно на відміну від кола, чи квадрату тут можна обійтись одним циклом, бо ромб не має абсолютно вертикальних, чи горизонтальних ліній. Побудову ромба здійснюємо за допомогою функції abs()(модуль числа).

*void setSide(float)* - метод, що визначає сторону ромба. В якості аргумента приймає одне число float;

*float getSide()* - метод, що повертає сторону ромба.

1. **Segment**

Класс Segment в своєму конструкторі містить визначення радіусу сегменту та куту. Також виклик метода, що змінює назву вікна, та метода, який змінює колір пікселів на синій.

Метод *draw()* для класу Segment працює подібно до кола, використовуючи також два цикли. Але діапазон даних визначається за допомогою знаходження висоти сегменту та довжини хорди. Таким чином, маючи кут та радіус ми можемо визначити, в якому діапазоні значень в декартовій системі нам відпрацювати цикл побудови кола.

*void setAngle(float)* - метод, що визначає кут сегменту. В якості аргумента приймає одне число float;

*void setRadius(float)* - метод, що визначає радіус сегменту. В якості аргумента приймає одне число float;

*float getAngle()* - метод, що повертає кут сегменту;

*float getRadius()* - метод, що повертає радіус сегменту.

## 2.2 Результати тестування програмного продукту.

Для демонстрації роботи класів створимо по одному об’єкту кожного з довільними даними:

| Line newLine(100.); Circle newCircle(50.); Square newSquare(50.); Rhombus newRhombus(35.); Segment newSegment(120., 50.); |
| --- |

Запускаємо програму і бачимо, що всі фігури будуються.

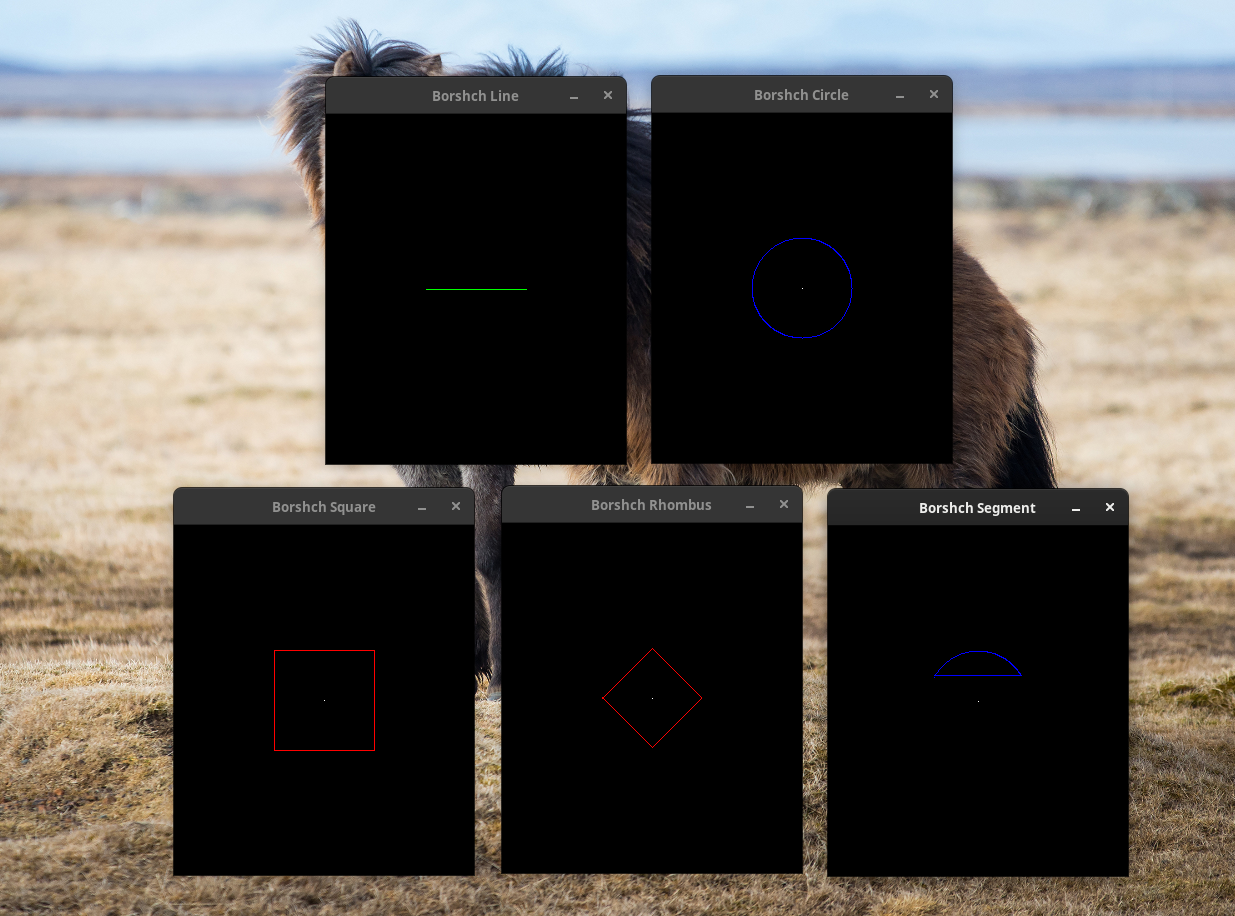


Рис 2.3 - Приклад роботи програми.

# ВИСНОВКИ

Як вже і було сказано, об’єктно-орієнтоване програмування є однією з основних парадигм програмування. С++ дуже потужний інструмент, який підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає в себе поширені контейнери і алгоритми, введення-виведення, регулярні вирази, підтримку багатопоточності і інші можливості. C++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником - мовою C - найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування.

В процесі виконання курсової роботи я підсумував свої, здобуті під час проходження курсу Технології безпечного програмування. Також ознайомився з бібліотекою SDL, яка буде корисною в майбутньому. За допомогою набутих навичок і знань створив просту програму, яка реалізує графічні примітиви та відображає їх на площині.

**Код** програми прикладаю в додатках А, Б та В. Також [тут](https://katb.in/akohuhugaqe) на сервісі Katbin.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проектів Вікімедіа. Об'єктно-орієнтоване програмування – Вікіпедія. Вікіпедія. URL: [https://uk.wikipedia.org/wiki/Об'єктно-орієнтоване\_програмування](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1'%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (дата звернення: 23.12.2021).
2. Учасники проектів Вікімедіа. Парадигма програмування – Вікіпедія. Вікіпедія. URL:

[https://uk.wikipedia.org/wiki/Парадигма\_програмування](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (дата звернення: 23.12.2021).

1. Учасники проектів Вікімедіа. C++ – Вікіпедія. Вікіпедія. URL: <https://uk.wikipedia.org/wiki/C++> (дата звернення: 23.12.2021).
2. FrontPage. FrontPage - SDL Wiki. URL: <https://wiki.libsdl.org/FrontPage> (date of access: 23.12.2021).

# ДОДАТКИ

## Додаток А

Код файлу main.cpp

| #include <iostream> #include <SDL2/SDL.h> #include "classes.cpp"  int main() {   SDL\_Event event;  SDL\_Init(SDL\_INIT\_VIDEO);   Line newLine(100.);  Circle newCircle(50.);  Square newSquare(50.);  Rhombus newRhombus(35.);  Segment newSegment(120., 50.);   newLine.draw();  newCircle.draw();  newSquare.draw();  newRhombus.draw();  newSegment.draw();   while (1) {  if (SDL\_PollEvent(&event) &&  (event.window.event == SDL\_WINDOWEVENT\_CLOSE || event.type == SDL\_QUIT))  break;  }   SDL\_Quit();  return EXIT\_SUCCESS;  } |
| --- |

## Додаток Б

Код файлу classes.cpp

| #include <SDL2/SDL.h> #include <iostream> #include <cmath> #include "const.cpp"  class Window {  protected:  SDL\_Renderer \*renderer;  SDL\_Window \*window;  public:  Window () {  SDL\_CreateWindowAndRenderer(SCREEN\_WIDTH, SCREEN\_HEIGHT, 0, &window, &renderer);  SDL\_SetRenderDrawColor(renderer, 0, 0, 0, 0);  SDL\_RenderClear(renderer);  SDL\_SetRenderDrawColor(renderer, 255, 255, 255, 255);  SDL\_RenderDrawPoint(renderer, SCREEN\_WIDTH/2, SCREEN\_HEIGHT/2);  }  ~Window (){  SDL\_DestroyRenderer(renderer);  SDL\_DestroyWindow(window);  } };   class Line : public Window {  private:  float length;  public:  Line(float newLength) {  length = newLength;  SDL\_SetWindowTitle(window, "Borshch Line");  SDL\_SetRenderDrawColor(renderer, 0, 255, 0, 255);  }   void setLength(float newLength) { length = newLength; }  float getLength() { return length; }   void draw() {  for (int i = SCREEN\_WIDTH/2-length/2; i <= SCREEN\_WIDTH/2+length/2; ++i) {  SDL\_RenderDrawPoint(renderer, i, SCREEN\_HEIGHT/2);  }  SDL\_RenderPresent(renderer);  } };   class Circle : public Window {  private:  float radius;  public:  Circle(float newRadius) {  radius = newRadius;  SDL\_SetWindowTitle(window, "Borshch Circle");  SDL\_SetRenderDrawColor(renderer, 0, 0, 255, 255);  }   void setRadius(float newRadius) { radius = newRadius; }  float getRadius() { return radius; }   void draw() {  for (int i = SCREEN\_WIDTH/2-radius; i <= SCREEN\_WIDTH/2+radius; ++i) {  SDL\_RenderDrawPoint(renderer, i, sqrt((radius\*radius)-((i-SCREEN\_WIDTH/2)\*(i-SCREEN\_WIDTH/2)))+SCREEN\_HEIGHT/2);  SDL\_RenderDrawPoint(renderer, i, -sqrt((radius\*radius)-((i-SCREEN\_WIDTH/2)\*(i-SCREEN\_WIDTH/2)))+SCREEN\_HEIGHT/2);  }  for (int i = SCREEN\_HEIGHT/2-radius; i <= SCREEN\_HEIGHT/2+radius; ++i) {  SDL\_RenderDrawPoint(renderer, sqrt((radius\*radius)-((i-SCREEN\_HEIGHT/2)\*(i-SCREEN\_HEIGHT/2)))+SCREEN\_WIDTH/2, i);  SDL\_RenderDrawPoint(renderer, -sqrt((radius\*radius)-((i-SCREEN\_HEIGHT/2)\*(i-SCREEN\_HEIGHT/2)))+SCREEN\_WIDTH/2, i);  }  SDL\_RenderPresent(renderer);  } };   class Square : public Window {  private:  float side;  public:  Square(float newSide) {  side = newSide;  SDL\_SetWindowTitle(window, "Borshch Square");  SDL\_SetRenderDrawColor(renderer, 255, 0, 0, 255);  }   void setSide(float newSide) { side = newSide; }  float getSide() { return side; }   void draw() {  for (int i = SCREEN\_WIDTH/2-side; i <= SCREEN\_WIDTH/2+side; ++i) {  SDL\_RenderDrawPoint(renderer, i, SCREEN\_HEIGHT/2+side);  SDL\_RenderDrawPoint(renderer, i, SCREEN\_HEIGHT/2-side);  }  for (int i = SCREEN\_HEIGHT/2-side; i <= SCREEN\_HEIGHT/2+side; ++i) {  SDL\_RenderDrawPoint(renderer, SCREEN\_WIDTH/2+side, i);  SDL\_RenderDrawPoint(renderer, SCREEN\_WIDTH/2-side, i);  }  SDL\_RenderPresent(renderer);  } };   class Rhombus : public Window {  private:  float xRange;  public:  Rhombus(float newSide) {  xRange = sqrt(2\*pow(newSide,2));  SDL\_SetWindowTitle(window, "Borshch Rhombus");  SDL\_SetRenderDrawColor(renderer, 255, 0, 0, 255);  }   void setSide(float newSide) { xRange = sqrt(2\*pow(newSide,2)); }  float getSide() { return sqrt(pow(xRange, 2)/2); }   void draw() {  for (int i = SCREEN\_WIDTH/2-xRange; i <= SCREEN\_WIDTH/2+xRange; ++i) {  SDL\_RenderDrawPoint(renderer, i, SCREEN\_HEIGHT/2+xRange+abs(i-SCREEN\_WIDTH/2)-xRange\*2);  SDL\_RenderDrawPoint(renderer, i, SCREEN\_HEIGHT/2+xRange-abs(i-SCREEN\_WIDTH/2));  }  SDL\_RenderPresent(renderer);  } };   class Segment : public Window {  private:  float angle, radius;  public:  Segment(float newAngle, float newRadius) {  angle = newAngle/57.296;  radius = newRadius;  SDL\_SetWindowTitle(window, "Borshch Segment");  SDL\_SetRenderDrawColor(renderer, 0, 0, 255, 255);  }   void setAngle(float newAngle) { angle = newAngle/57.296; }  void setRadius(float newRadius) { radius = newRadius; }  float getAngle() { return angle; }  float getRadius() { return radius; }   void draw() {  for (int i = SCREEN\_WIDTH/2-(2\*radius\*sin(angle/2))/2; i <= SCREEN\_WIDTH/2+(2\*radius\*sin(angle/2))/2; ++i) {  SDL\_RenderDrawPoint(renderer, i, -sqrt((radius\*radius)-((i-SCREEN\_WIDTH/2)\*(i-SCREEN\_WIDTH/2)))+SCREEN\_HEIGHT/2);  SDL\_RenderDrawPoint(renderer, i, SCREEN\_HEIGHT/2-radius+(radius\*(1-cos(angle/2))));  }  for (int i = SCREEN\_HEIGHT/2-radius; i <= SCREEN\_HEIGHT/2-radius+(radius\*(1-cos(angle/2))); ++i) {  SDL\_RenderDrawPoint(renderer, sqrt((radius\*radius)-((i-SCREEN\_HEIGHT/2)\*(i-SCREEN\_HEIGHT/2)))+SCREEN\_WIDTH/2, i);  SDL\_RenderDrawPoint(renderer, -sqrt((radius\*radius)-((i-SCREEN\_HEIGHT/2)\*(i-SCREEN\_HEIGHT/2)))+SCREEN\_WIDTH/2, i);  }  SDL\_RenderPresent(renderer);  } }; |
| --- |

## Додаток В

Код const.cpp та Makefile

| //const.cpp  const int SCREEN\_WIDTH = 300; const int SCREEN\_HEIGHT = 350; |
| --- |

| //Makefile  CC=g++  CFLAGS=-c -Wall  all: course  course: main.o  $(CC) main.o -o course -I/usr/local/include/SDL2 -L/usr/local/lib -lSDL2  main.o: main.cpp const.cpp classes.cpp  $(CC) $(CFLAGS) main.cpp const.cpp classes.cpp  clean:  rm -rf \*.o course |
| --- |