rr

**МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**Лабораторна робота № 3**

**з дисципліни “ Основи програмування ”**

**тема “ЦИКЛІЧНІ КОНСТРУКЦІЇ”**

**.**

|  |  |  |
| --- | --- | --- |
| **Виконав**  **студент I курсу**  **групи КП-83**  **Мортіков Владислав Євгенович**  **(прізвище, ім’я, по батькові)**  **варіант № 38** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **Гадиняк Руслан Анатолійович**  **(прізвище, ім’я, по батькові)** |

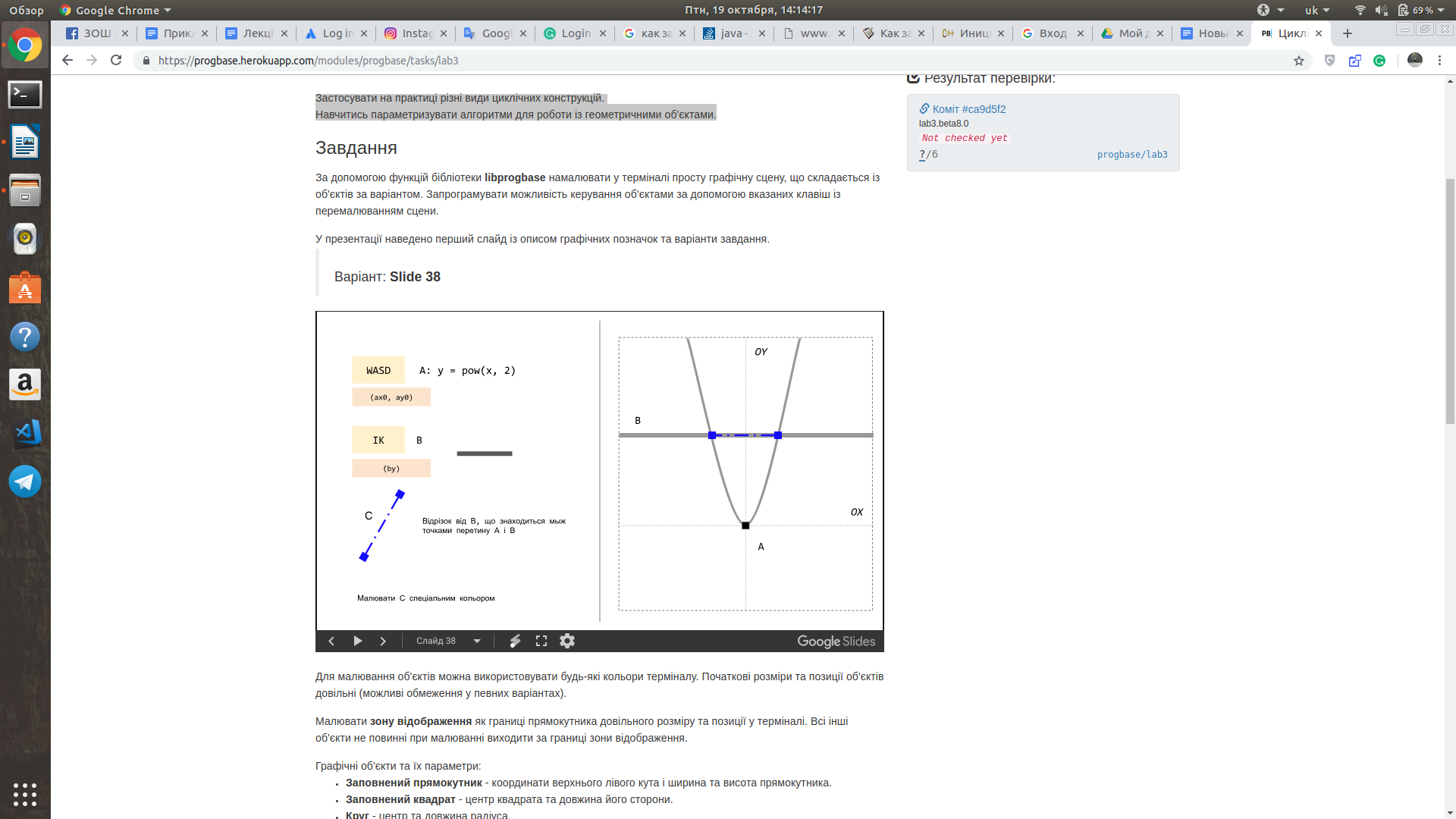
**Київ 2018**

**Мета роботи**

Застосувати на практиці різні види циклічних конструкцій та навчитись параметризувати алгоритми для роботи із геометричними об'єктами.

**Постановка завдання**

За допомогою функцій бібліотеки **libprogbase** намалювати у терміналі просту графічну сцену, що складається із об'єктів за варіантом. Запрограмувати можливість керування об'єктами за допомогою вказаних клавіш із перемалюванням сцени.



Для малювання об'єктів можна використовувати будь-які кольори терміналу. Початкові розміри та позиції об'єктів довільні (можливі обмеження у певних варіантах).

Малювати зону відображення як границі прямокутника довільного розміру та позиції у терміналі. Всі інші об'єкти не повинні при малюванні виходити за границі зони відображення.

Графічні об'єкти та їх параметри:

* Відрізок - координати кінців відрізку.
* Графік функції - точка початку координат цієї функції і сама функція.

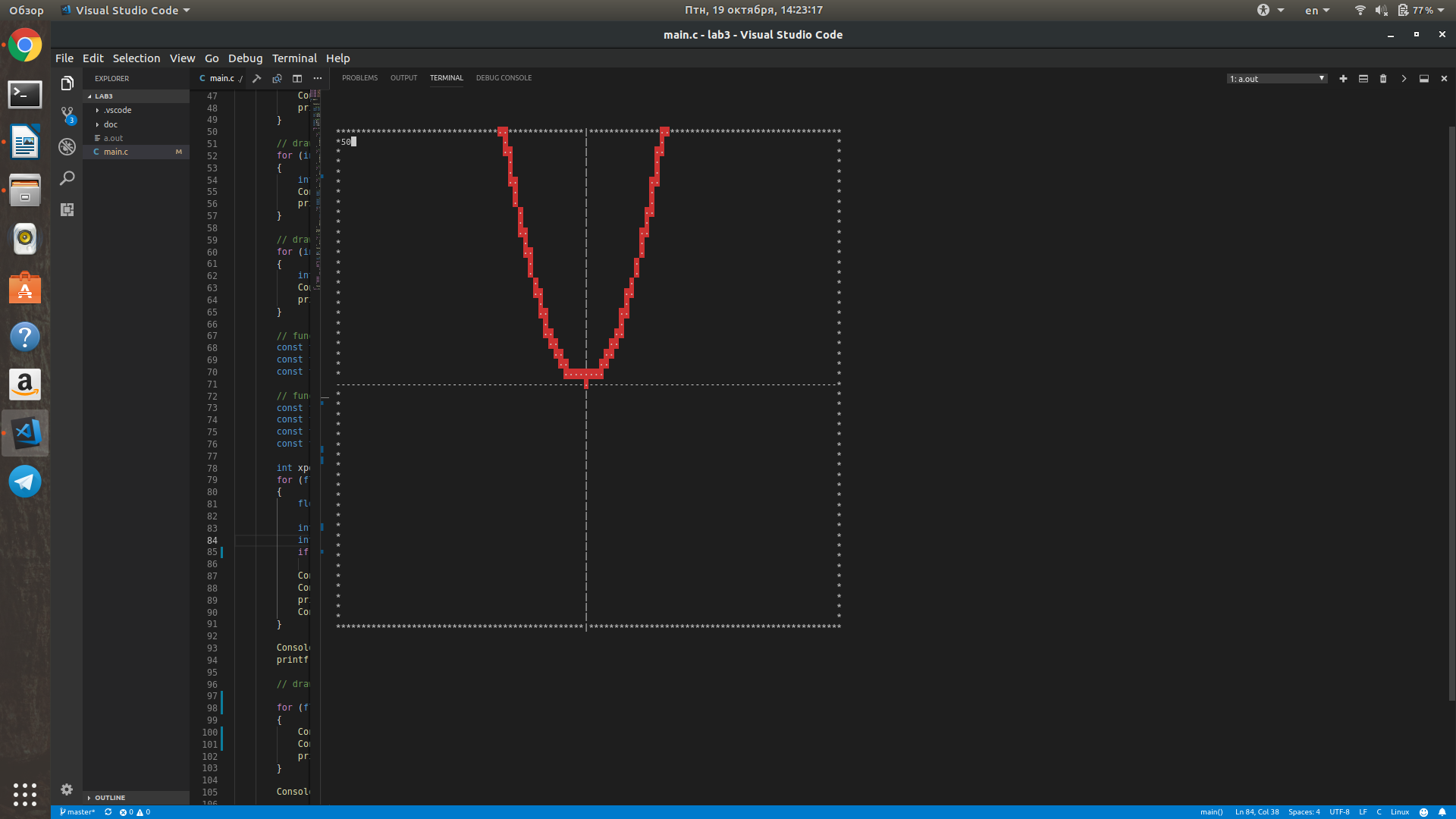
Для малювання різних геометричних фігур і графіків можна опиратись на код із практики.

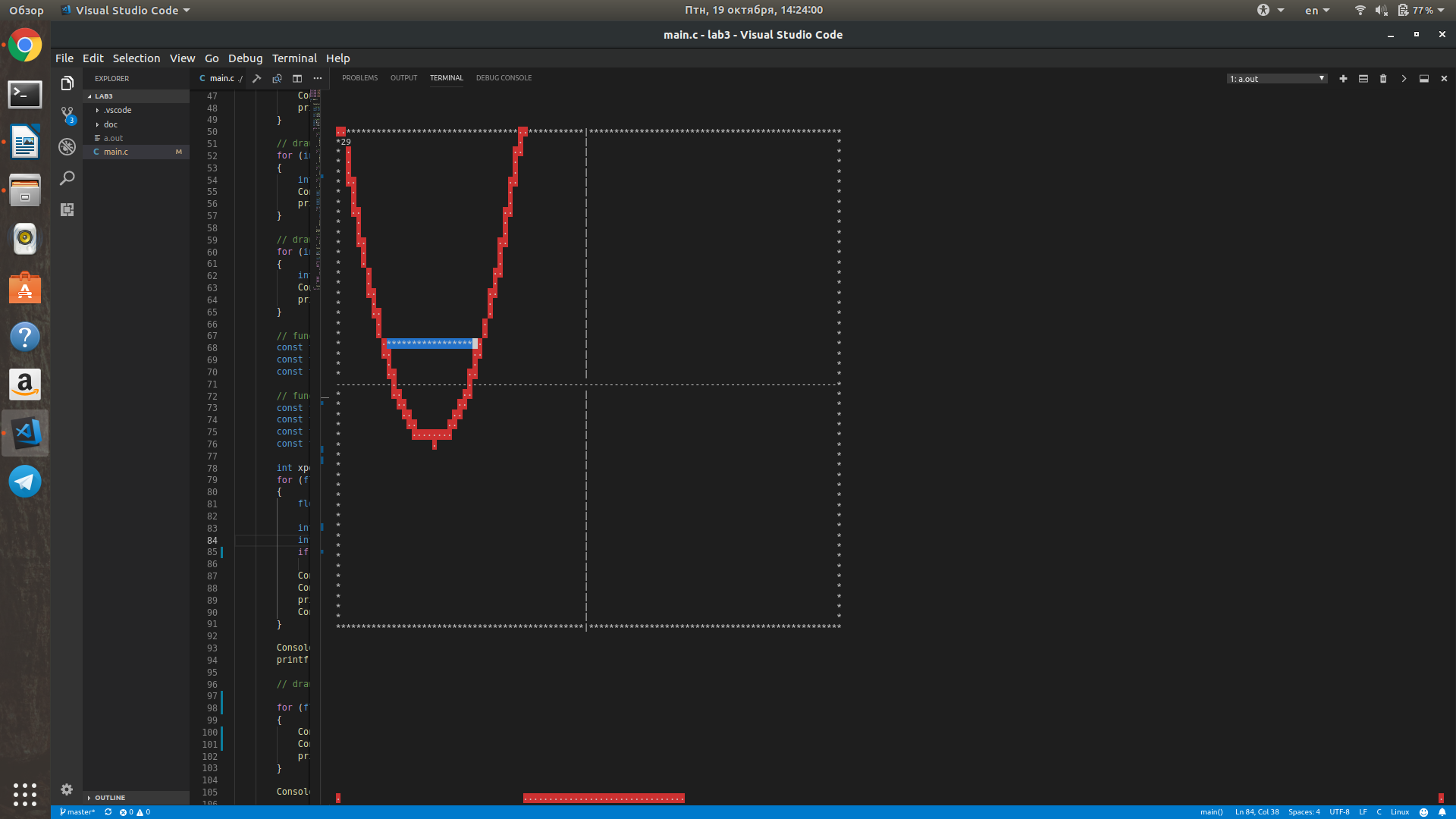
Для малювання відрізку між двома точками використовувати *універсальний алгоритм Брезенхема*.

**Текст коду програми**

|  |
| --- |
| **main.c** |
| #include <stdio.h>  #include <math.h>  #include <stdlib.h>  #include <progbase.h>  #include <progbase/console.h>  int main()  {  Console\_clear();  // border size  const int width\_rz = 100;  const int height\_rz = 50;  // origin point shift  int centerx = width\_rz / 2;  int centery = height\_rz / 2 + 1;  // graphic coord.  int grx = width\_rz / 2;  int gry = centery;  int line\_ab\_y = centery;  int line\_ab\_x = centerx;  while (1)  {  // draw border (horizontal lines)  for (int border\_w = 1; border\_w <= width\_rz; border\_w = border\_w + 1)  {  int border\_h = 1;  Console\_setCursorPosition(border\_h, border\_w);  printf("\*");  border\_h = height\_rz;  Console\_setCursorPosition(border\_h, border\_w);  printf("\*");  }  // draw border (vertical lines)  for (int border\_h = 1; border\_h <= height\_rz; border\_h = border\_h + 1)  {  int border\_w = 1;  Console\_setCursorPosition(border\_h, border\_w);  printf("\*");  border\_w = width\_rz;  Console\_setCursorPosition(border\_h, border\_w);  printf("\*");  }  // draw y line  for (int yline = 1; yline <= height\_rz; yline++)  {  int xline = centerx;  Console\_setCursorPosition(yline, xline);  printf("|");  }  // draw x line  for (int xline = 1; xline < width\_rz; xline++)  {  int yline = centery;  Console\_setCursorPosition(yline, xline);  printf("-");  }  // function x  const float xmin = -5;  const float xmax = 5;  const float xstep = 0.005;  // function-to-console transformation values  const float xwidth = xmax - xmin;  const float yheight = 5;  const float ax0 = width\_rz / xwidth;  const float ay0 = height\_rz / yheight;  int xper = 0;  for (float x = xmin; x <= xmax; x = x + xstep)  {  float y = pow(x, 2);  int gx = grx + ax0 \* x;  int gy = gry - (ay0 \* y);  if (gy == line\_ab\_y)  xper = gx;  Console\_setCursorAttribute(BG\_RED);  Console\_setCursorPosition(gy, gx);  printf("·");  Console\_reset();  }  Console\_setCursorPosition(2, 2);  printf("%d", xper);  // draw line AB    for (float x = line\_ab\_x - xper; x < xper - line\_ab\_x - 1; x++)  {  Console\_setCursorAttribute(BG\_BLUE);  Console\_setCursorPosition(line\_ab\_y, line\_ab\_x + x);  printf("\*");  }  Console\_reset();  // user key input  char keyCode = 0;  keyCode = Console\_getChar();  // controls  const int tStep = 1;  switch (keyCode)  {  case 'w':  if (gry >= 10)  {  gry -= tStep;  }  break;  case 'a':  if (grx > 20)  { line\_ab\_x -= tStep;  grx -= tStep;  }  break;  case 's':  if (gry < 49)  {  gry += tStep;  }  break;  case 'd':  if (grx < width\_rz-20)  {  line\_ab\_x += tStep;  grx += tStep;  }  break;  case 'i':  if (line\_ab\_y > 1)  {  line\_ab\_y -= tStep;  }  break;  case 'k':  line\_ab\_y += tStep;  break;  }  Console\_clear();  }  return 0;  } |

**￼Приклади результатів**

****

****

**Висновки**

Виконавши дану лабораторну роботу було застосовано на практиці різні види циклічних конструкцій. Навчились параметризувати алгоритми для роботи із геометричними об'єктами.

Побудовано параболу, між вітками якої пересувається пряма, збільшуючись або зменшуючись, відповідно до розмірів і положення точок параболи в даний момент часу.

Парабола, у межах можливого, не виходить за кордони робочої зони.

Компіляція всього коду відбувалася за допомогою компілятора gcc.