

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 №** 6

з дисципліни “ Основи програмування ”

тема “Структури даних”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) I курсу  групи КП-83  Клименко Ярослав Олександрович  (*прізвище, ім’я, по батькові*)    варіант № 8 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

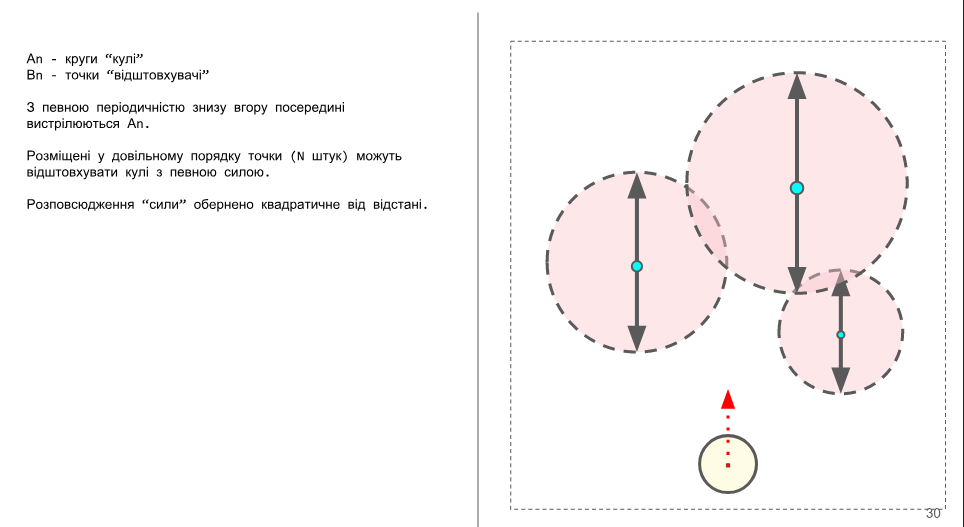
Київ 2018

**Мета роботи**

Навчитися працювати зі структурами даних.

**Постановка завдання**

За допомогою функцій модуля **Canvas** бібліотеки **libprogbase** (progbase/canvas.h, версія бібліотеки не менше [0.6.0](https://github.com/PublicHadyniak/libprogbase/releases/tag/0.6.0)) намалювати у терміналі графічну сцену, що складається із графічних об'єктів та піддається певним правилам взаємодії за варіантом.

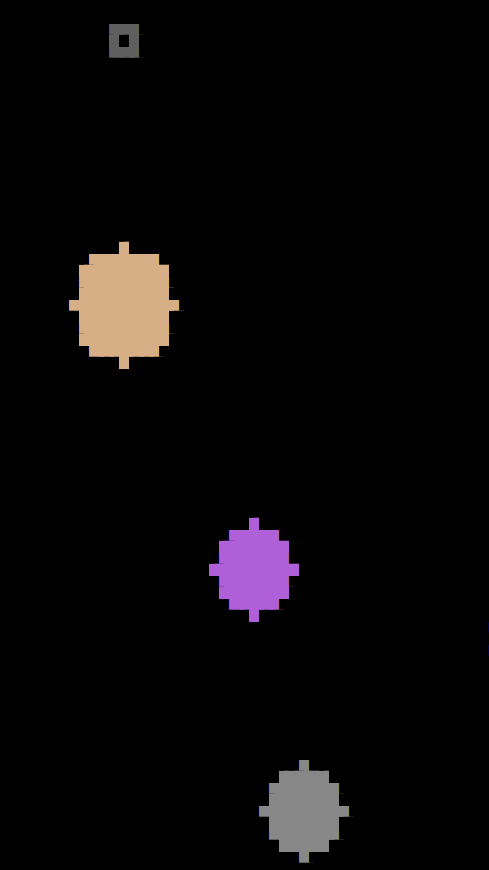
Виділити відповідні типи даних графічних об'єктів як структури і використовувати змінні цих структур для зберігання їх параметрів. Однотипні змінні об'єднати у масиви структур.

**Тексти коду програм**

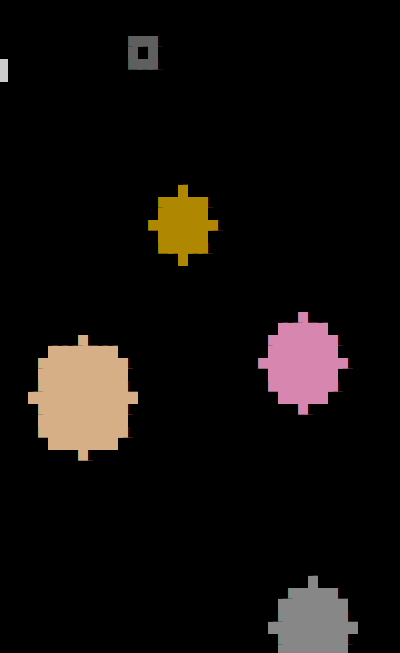
|  |
| --- |
| **main.c** |
| **// Компілювати за допомогою:**  **// gcc main.c -lprogbase -lm**  **#include <stdio.h> // Для друку в термінал**  **#include <math.h> // Для математичних функцій**  **#include <stdlib.h> // Деякі додаткові функції**  **#include <progbase.h>**  **#include <time.h>**  **#include <progbase/console.h>**  **#include <progbase/canvas.h> // Спрощений ввід і вивід даних у консоль**  **struct cords**  **{**  **float x;**  **float y;**  **};**  **struct vect**  **{**  **float x;**  **float y;**  **};**  **struct color**  **{**  **int red;**  **int green;**  **int blue;**  **};**  **struct Ball**  **{**  **struct cords loc;**  **struct color bColor;**  **int radius;**  **struct vect speed;**  **};**  **struct point**  **{**  **struct cords loc;**  **struct color color;**  **int radius;**  **float power;**  **struct vect distance;**  **};**  **int randRGB();**  **int main()**  **{**  **Console\_clear();**  **srand(time(0));**  **struct ConsoleSize cs = Console\_size();**    **const int w = cs.columns;**  **const int h = cs.rows \* 2;**  **const int delay = 100;**  **const int gravitation = 2;**  **const int pointAverage = rand() % (5 - 1 + 1) + 1;**  **const int ballAverage = 5;**  **struct point points[pointAverage];**  **for (int i = 0; i < pointAverage; i++)**  **{**  **points[i].radius = 2;**  **points[i].loc.x = rand() % (w - points[i].radius + 1 - 10) + 10;**  **points[i].loc.y = rand() % (h - points[i].radius + 1 - 10) + 10;**  **points[i].color.red = randRGB();**  **points[i].color.green = randRGB();**  **points[i].color.blue = randRGB();**  **points[i].power = 0;**  **}**  **struct Ball ball[ballAverage];**  **for (int i = 0; i < ballAverage; i++)**  **{**  **ball[i].radius = rand() % (5 + 1 - 3) + 3;**  **ball[i].loc.x = w / 2;**  **ball[i].loc.y = ball[i].radius + 1;**  **ball[i].bColor.red = randRGB();**  **ball[i].bColor.green = randRGB();**  **ball[i].bColor.blue = randRGB();**  **ball[i].speed.x = 0;**  **ball[i].speed.y = 3;**  **};**  **int lich = 0;**  **Canvas\_setSize(w, h);**  **Canvas\_invertYOrientation();**  **do**  **{**  **lich++;**  **for (int i = 0; i < lich / 10; i++)**  **{**  **if( i == ballAverage){**  **break;**  **}**  **for (int j = 0; j < pointAverage; j++)**  **{**  **points[j].distance.x = ball[i].loc.x - points[j].loc.x;**  **points[j].distance.y = ball[i].loc.y - points[j].loc.y;**  **points[j].power = sqrt(pow(points[j].distance.x, 2) + pow(points[j].distance.y, 2));**  **if(points[j].loc.x > ball[i].loc.x){**  **ball[i].speed.x += points[j].distance.x \* gravitation/ pow(points[j].power, 2);**  **}**  **if(points[j].loc.x < ball[i].loc.x){**  **ball[i].speed.x -= points[j].distance.x \* gravitation / pow(points[j].power, 2);**  **}**  **if(points[j].loc.y > ball[i].loc.y){**  **ball[i].speed.y += points[j].distance.y \* gravitation / pow(points[j].power, 2);**  **}**  **if(points[j].loc.y < ball[i].loc.y){**  **ball[i].speed.y -= points[j].distance.y \* gravitation / pow(points[j].power, 2);**  **}**  **}**  **if (ball[i].loc.x + ball[i].radius >= w - 1 || ball[i].loc.x - ball[i].radius <= 1)**  **{**  **ball[i].speed.x = -ball[i].speed.x;**  **}**  **if (ball[i].loc.y + ball[i].radius >= h - 1 || ball[i].loc.y - ball[i].radius <= 1)**  **{**  **ball[i].speed.y = -ball[i].speed.y;**  **}**  **if (ball[i].speed.x == 0 || ball[i].speed.y == 0)**  **{**  **ball[i].speed.x += 3;**  **ball[i].speed.y += 3;**  **}**  **ball[i].loc.x += ball[i].speed.x;**  **ball[i].loc.y += ball[i].speed.y;**    **}**  **Canvas\_beginDraw();**  **for (int i = 0; i <= lich; i += 10)**  **{**  **if(i / 10 == ballAverage){**  **break;**  **}**  **Canvas\_setColorRGB(ball[(i / 10)].bColor.red, ball[i / 10].bColor.green, ball[i / 10].bColor.blue);**  **Canvas\_fillCircle(ball[i / 10].loc.x, ball[i / 10].loc.y, ball[i / 10].radius);**    **}**  **for (int k = 0; k < pointAverage; k++)**  **{**  **Canvas\_setColorRGB(points[k].color.red, points[k].color.green, points[k].color.blue);**  **Canvas\_strokeCircle(points[k].loc.x, points[k].loc.y, 2);**  **}**    **Canvas\_endDraw();**  **sleepMillis(delay);**    **} while (!Console\_isKeyDown());**  **return 0;**  **}**  **int randRGB()**  **{**  **int a = (rand() % 255 + 1);**  **return a;**  **}** |

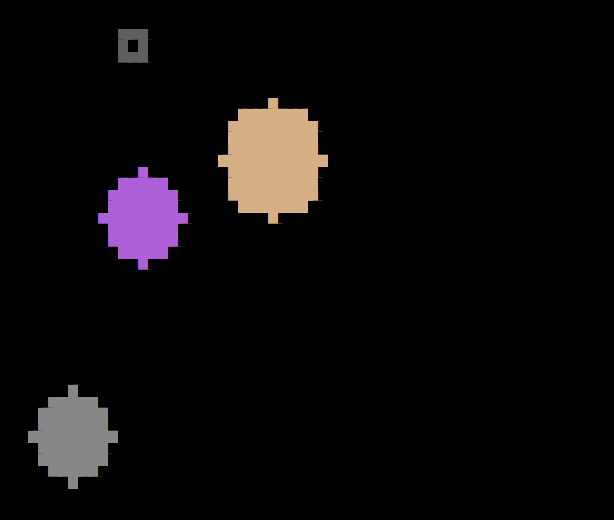
**Приклади результатів**

Початок роботи:



Зміна траекторії польоту кулі:





**Висновки**

Під час виконання даної лабораторної роботи, я ознайомився з принципами роботи із структурами даних та їх обробкою у мові програмування С.

Реалізація програми відбулася за допомогою вкладених циклів, а також структур даних. При виконанні завдання, я користувався бібліотекою для мови С - libprogbase.h та її функціями, що допомогли спростити процес малювання фігур. Реалізація переміщення об”єктів відбулася за допомогою векторів та структур даних, що допомогли ними керувати.

Компіляція всього коду відбувалася за допомогою компілятора gcc.