**2015**

**Графічні можливості мови програмування PascalABC**

**Мова програмування pascalabc**

рябошапка в.о.

# **Зміст**

[Зміст 1](#_Toc440628282)

[Графічні можливості мови програмування PascalABC. 2](#_Toc440628283)

[Основні поняття 2](#_Toc440628284)

[**Графічні команди модуля GraphABC для роботи з олівцем та пензлем.** 3](#_Toc440628285)

[**Константи cl, які задають колір** 3](#_Toc440628286)

[**Константи bs, які задають стиль пензля** 3](#_Toc440628287)

[**Команди для керування графічним вікном** 4](#_Toc440628288)

[**Команди для створення графічних примітивів модуля GraphABC** 4](#_Toc440628289)

[**Команди для роботи з шрифтами** 5](#_Toc440628290)

[**Константи fs, які задають стиль шрифту** 5](#_Toc440628291)

[**Команди для роботи з растровими графічними зображеннями** 5](#_Toc440628292)

[Приклади роботи з графічними зображеннями 6](#_Toc440628293)

[**Приклад 1.** 6](#_Toc440628294)

[Приклади використання математичних функцій до побудови графічних зображень 8](#_Toc440628295)

[Побудова графіків та діаграм 10](#_Toc440628296)

[**Завдання** 11](#_Toc440628297)

[Обробка подій в графічному вікні 13](#_Toc440628298)

[**Віртуальні коди клавіш** 14](#_Toc440628299)

[**Приклад програми для обробки подій.** 14](#_Toc440628300)

# **Графічні можливості мови програмування PascalABC.**

## **Основні поняття**

Зображення на екрані монітора формується з точок, які називають пікселями. Кожна точка має свої властивості, які визначають її положення на екрані (координати) та колір. В якості координат виступають порядкові номера пікселів по горизонталі та вертикалі, координати задаються тільки цілими числами. Початок координат знаходиться в лівому верхньому куті екрана. Значення горизонтальної координати (абсциси) відраховується зліва на право, а значення вертикальної координати (ординати) – зверху вниз.

***х***

***у***

***(0;0)***

***(w;0)***

***(0;h)***

***(w;h)***

***(x;y)***

При створенні малюнка засобами мови програмування PascalABC формується векторне зображення, яке складається з простих графічних фігур – так званих графічних примітивів (точка, відрізок, прямокутник, коло, овал).

Для створення графічного об'єкта потрібно підключити модуль в якому описано всі команди для створення і обробки графічних зображень. Всі ці команди знаходяться в спеціальній бібліотеці, яка має назву **GraphABC**. Дана бібліотека підключається в описовій частині програми в розділі а **Uses**.

При створені графічного зображення перше, що потрібно зробити, це задати розміри графічного вікна в якому буде виведено зображення. Розміри вікна задаються відповідною командою **SetWindowSize(w,h)**, яка містить в собі максимальне значення пікселів по горизонталі – **w** та вертикалі – **h**.

Основними інструментами створення графічних об'єктів є олівець та пензель, які позначаються певними командами модуля **GraphABC**. За допомогою олівця малюють ліній та контури фігур, за допомогою пензля відбувається фарбування обмеженої лініями області графічного вікна.

### **Графічні команди модуля GraphABC для роботи з олівцем та пензлем.**

|  |  |
| --- | --- |
| Команда | Призначення команди |
| **setPenWidth(n);** | Встановлює товщину олівця в пікселях, де n – ціле число |
| **setPenColor(cl);** | Встановлює колір олівця, де cl – константа, яка задає колір |
| **SetPenStyle(ps);** | Встановлює стиль лінії олівця, де ps – ціле число від 0 до 4 |
| **SetBrushColor(cl);** | Встановлює колір пензля, де cl – константа, яка задає колір |
| **SetBrushStyle(bs);** | Встановлює стиль пензля, де bs – константа, яка задає стиль |
| **SetBrushPicture (fname);** | Встановлює в якості зразка для зафарбовування пензлем графічне зображення, яке зберігається в файлі з назвою **fname** |

При роботі з графічним вікном, для виведення в нього текстових повідомлень необхідно підключити модуль **CRT**. Даний модуль містить команди, які дають можливість виводити текстові повідомлень в графічному вікні. Модуль підключається в розділі **Uses** разом з модулем **GraphABC**, які записуються через кому за необхідності також в даному розділі підключаються інші модулі, навіть створені користувачем. Замість константи **cl**, яка задає колір можна використовувати процедуру **RGB(r,g,b)**, де **r,g,b** – числові значення стандартних кольорів (червоний, зелений, синій) від 0 до 255.

### **Константи cl, які задають колір**

|  |  |  |  |
| --- | --- | --- | --- |
| Константа | Колір | Константа | Колір |
| **clBlack** | чорний | **clAqua** | бірюзовий |
| **clPurple** | фіолетовий | **clCream** | кремовий |
| **clWhite** | білий | **clOlive** | оливковий |
| **clMaroon** | темно-червоний | **clFuchsia** | бузковий |
| **clRed** | червоний | **clTeal** | синє-зелений |
| **clNavy** | темно-синій | **clGray** | зелений |
| **clGreen** | зелений | **clLime** | світло-зелений |
| **clBrown** | коричневий | **clLtGray** | світло-сірий |
| **clBlue** | синій | **clDarkGray** | темно-сірий |
| **clSkyBlue** | голубий | **clMedGray** | сірий |
| **clYellow** | жовтий | **clSilver** | серебряний |

### **Константи bs, які задають стиль пензля**

|  |  |  |  |
| --- | --- | --- | --- |
| Константа | Стиль | Константа | Стиль |
| **bsSolid** | суцільне заповнення вибраним кольором | **bsCross** | заповнення клітинками |
| **bsClear** | заповнення прозорим кольором | **bsDiagCross** | заповнення діагональними клітинками |
| **bsHorizontal** | заповнення горизонтальними лініями | **bsBDiagonal** | заповнення діагональними лініями // |
| **bsVertical** | заповнення вертикальними лініями | **bsFDiagonal** | Заповнення діагональними лініями \\ |

### **Команди для керування графічним вікном**

|  |  |
| --- | --- |
| Команда | Дія |
| **SetWindowSize(w,h:integer);** | Задає ширину та висоту графічного вікна |
| **SetWindowCaption(s: string);** | Встановлює назву графічного вікна, яке задається змінною **s** |
| **SaveWindow(fname: string);** | Зберігає вміст графічного вікна в файл із назвою **fname** |
| **LoadWindow(fname:**  **string);** | Завантажує в графічне вікно малюнок, який зберігається в файлі з назвою **fname** (формати файлів повинні бути **bmp**, **jpg** або **gif**), назва файлу заключається в одинарні лапки |
| **ClearWindow;** | Очищує графічне вікно і встановлює білий фон вікна |
| **ClearWindow(c:**  **ColorType);** | Очищує графічне вікно і встановлює фон вікна заданим кольором **с** |
| **Redraw;** | Перерисовує вміст графічного вікна |

### **Команди для створення графічних примітивів модуля GraphABC**

|  |  |
| --- | --- |
| Команда | Дія |
| **SetPixel(x,y,color:**  **integer);** | Зафарбовує один піксель з координатами (**х,у**) кольором **color** |
| **MoveTo(x,y: integer);** | Переміщує невидимий олівець в точку з координатами (**х,у**); працює в парі з командою LineTo(x,y); |
| **LineTo(x,y: integer);** | Малює лінію від поточних координат олівця до точки з координатами (**х,у**), при цьому координати олівця стають рівними (**х,у**) |
| **Line(x1,y1,x2,y2:**  **integer);** | Малює відрізок з початком в точці з координатами (**х1,у1**) та кінцем в точці з координатами (**х2,у2**) |
| **Circle(x,y,r: integer);** | Малює коло з центром в точці з координатами (**х,у**) та радіусом **r** – одиницею вимірювання радіуса є піксель |
| **Ellipse(x1,y1,x2,y2:**  **integer);** | Малює овал який вписаний в невидимий прямокутник, де (**х1,у1**) та (**х2,у2**) – координати початку та кінця діагоналі прямокутника |
| **Rectangle(x1,y1,x2,y2:**  **integer);** | Малює прямокутник, де (**х1,у1**) та (**х2,у2**) – координати початку та кінця діагоналі прямокутника |
| **Arc(x,y,r,a1,a2:**  **integer);** | Малює дугу кола з центром в точці з координатами (**х,у**) та радіусом **r,** яка обмежена двома променями, що утворюють кути **а1** та **а2** з віссю абсцис ОХ, кути задаються в градусах, відлік відбувається проти годинникової стрілки |
| **Pie(x,y,r,a1,a2:**  **integer);** | Малює сектор круга обмежений дугою кола з координатами (**х,у**) та радіусом **r,** яка обмежена двома променями, що утворюють кути **а1** та **а2** з віссю абсцис ОХ, кути задаються в градусах, відлік відбувається проти годинникової стрілки |
| **FloodFill(x,y,color:**  **integer);** | Зафарбовує обмежену лініями область графічного зображення вказаним кольором **color,** починаючи з точки з координатами (**х,у**), яка повинна знаходитись в середині зафарбовуваної області |
| **TextOut(x,y: integer; s:string);** | Виводить рядок символів **s** з точки (**х,у**), точка (**х,у**) – задає координати лівого верхнього кута прямокутника в якому виводиться рядок символів |

### **Команди для роботи з шрифтами**

|  |  |
| --- | --- |
| Команда | Дія |
| **SetFontColor(cl);** | Встановлює колір олівця, де cl – константа, яка задає колір |
| **SetFontSize(sz: integer);** | Встановлює розміри шрифту в пунктах, **sz –** цілочисленне значення |
| **SetFontName(shf: string);** | Задає шрифт, де **shf –** назва шрифту (MS Sans Serif – шрифт який використовується за замовчуванням; Times, Arial та Courier New – шрифти, які найчастіше використовують) |
| **SetFontStyle(fs);** | Встановлює стиль шрифту, де **fs–**константа, яка задає стиль |

### **Константи fs, які задають стиль шрифту**

|  |  |  |  |
| --- | --- | --- | --- |
| Константа | Стиль | Константа | Стиль |
| **fsNormal** | звичайний | **fsUnderline** | підкреслений |
| **fsBold** | жирний | **fsBoldUnderline** | жирний підкреслений |
| **fsItalic** | курсив | **fsItalicUnderline** | підкреслений курсив |
| **fsBoldItalic** | жирний курсив | **fsBoldItalicUnderline** | жирний підкреслений курсив |

### **Команди для роботи з растровими графічними зображеннями**

|  |  |
| --- | --- |
| Команда | Дія |
| **LoadPicture(fname:**  **string): integer;** | Завантажує в пам'ять комп'ютера малюнок, який зберігається в файлі з назвою **fname** (формати файлів повинні бути **bmp**, **jpg, png** або **gif**) та присвоює його деякому цілочисленному ідентифікатору **m**: **m:=LoadPicture(fname);** назва файлу заключається в одинарні лапки |
| **DrawPicture(m,x,y:**  **integer);** | Виводить завантажений графічний файл, який присвоєний ідентифікатору **m** в позицію з координатами (**х,у**) – верхній лівий кут прямокутної області в якій виводиться зображення завантажене з графічного файлу |
| **DrawPicture(m,x,y,w,h:**  **integer);** | Виводить завантажений графічний файл, який присвоєний ідентифікатору **m** в позицію з координатами (**х,у**) – верхній лівий кут прямокутної області в якій виводиться зображення завантажене з графічного файлу, **w, h –** відповідно ширина та висота прямокутної області в пікселях, малюнок масштабується для виведення в заданій області |
| **SavePicture(m: integer;**  **fname:string);** | Зберігає малюнок, який зв'язаний з ідентифікатором **m** в файл, ім'я якого представлено рядком символів **fname** (формати файлів мають бути **bmp**, **jpg** або **gif**) |
| **SetPictureSize(m,w,h:**  **integer);** | Встановлює розміри малюнка зв'язаного з ідентифікатором **m, w** – ширина та **h** – висота відповідно в пікселях |
| **SetPictureTransparent (m:integer; b: boolean);** | Встановлює якщо (**b=True**) або вимикає коли (**b=False**) режим прозорості при виведенні в графічному вікні малюнка зв'язаного з ідентифікатором **m** |

## **Приклади роботи з графічними зображеннями**

**Приклад 1.** Вивести в графічному вікні розмірами 640 на 480 пікселів, зображення основних графічних фігур (прямокутник, коло, овал, сектор) трикутника, дугу кінці якої з'єднані відрізком та зафарбувати їх різними кольорами та стилями. Над фігурами вивести напис "Геометричні фігури в PascalABC" та вивести даний напис в заголовку графічного вікна.

program qpirypu;

uses graphABC;{підключення графічного модуля}

begin

setWindowSize(640,480); {задається графічне вікно та його розміри}

SetWindowCaption('Геометричні фігури в PascslABC'); {встановлює назву графічного вікна}

setFontSize(15); {розміри шрифтa}

setFontStyle(fsBold); {встановлює напівжирний стиль шрифтa}

setFontColor(clBrown); {встановлює колір шрифта}

setBrushColor(clLtGray); {встановлює колір фону на якому виводиться текст}

textOut(130,10,'Гeометричні фігури в PascslABC'); {виведення тексту}

setPenWidth(3); {товщина олівця}

setPenColor(clBrown); {колір олівця}

setBrushColor(clYellow); {колір пензля}

SetBrushStyle(bsSolid); {встановлює стиль зафарбовування – суцільний}

rectangle(40,80,200,160); {прямокутник}

setPenColor(clRed); setBrushColor(clAqua);

SetBrushStyle(bsClear); {встановлює стиль зафарбовування – прозорий}

circle(300,120,40); {круг}

setPenColor(clBlue); setBrushColor(clRed);

SetBrushStyle(bsHorizontal);{встановлює стиль зафарбовування – горизонтальні лінії}

ellipse(400,80,540,160); {овал}

setPenColor(clFuchsia); setBrushColor(clLime);

SetBrushStyle(bsVertical); {встановлює стиль зафарбовування – вертикальні лінії}

Pie(100,300,100,45,130); {сектор}

setPenColor(clSilver);

SetBrushStyle(bsCross); {встановлює стиль зафарбовування – клітинами}

Arc(300,300,100,45,190); {дуга}

Line(369,229,203,316); {відрізок}

FloodFill(220,290,clOlive); {зафарбовує обмежену дугою та відрізком область}

setPenColor(clNavy);

SetBrushStyle(bsBDiagonal);{встановлює стиль зафарбовування – діагональними лініями}

MoveTo(400,310); {встановлює графічний курсор в точку з вказаними координатами}

LineTo(450,200); {малює лінію з останньої позиції курсору до точки з координатами}

LineTo(500,310); {малює лінію з останньої позиції курсору до точки з координатами}

LineTo(400,310); {малює лінію з останньої позиції курсору до точки з координатами}

FloodFill(450,300,clSkyBlue); {зафарбовує обмежену відрізками область}

end.

## **Приклади використання математичних функцій до побудови графічних зображень**

В нижче наведеній програмі будується астроїда за допомогою множини еліпсів, які до неї дотикаються. Еліпси в програмі будуються по точках координати яких обчислюються за допомогою параметричних рівняньеліпсадекартової системи координат:

Program astroida;

Uses GraphABC;

Const a=320; b=210;

Var s:real; c,m,n,x,y:integer;

begin

for m:=0 to 250 do

begin

s:=pi\*m/(2\*250);

for n:=0 to 80 do

begin

x:=Trunc(a\*(80-n)\*cos(s)/80);

y:=Trunc(b\*n\*sin(s)/80);

setpixel(a+x, b+y, clFuchsia);

setpixel(a+x, b-y, clFuchsia);

setpixel(a-x, b+y, clFuchsia);

setpixel(a-x, b-y, clFuchsia);

end;

end;

end.

В наступній програмі будується просторова крива "гелікоїд". Координати точок даної кривої в наведеній нижче програмі обчислюються за допомогою параметричних рівнянь просторової системи координат:

Program bokal;

Uses GraphABC;

Const a=400;

b=470;

c=380;

d=10;

m=5000;

Vart,r,s:real;

i,i1,i2,x,y,u:integer;

Begin

SetWindowSize(800,900);

For i:=550 to m do

Begin

t:=i/m;

r:=(ln(30\*t+6)-ln(20))/ln(4);

u:=Trunc(100/t);

For i1:=0 to 40 do

For i2:=1 to 3 do

begin

s:=Pi/2\*(t+i1/8+i2/24);

x:=Trunc(c\*r\*cos(s));

y:=Trunc(d\*r\*sin(s));

SetPixel(a+x,2\*b-d+y-u,clMoneyGreen);

end;

end;

end.

## **Побудова графіків та діаграм**

При вивченні різних явищ природи та людської спільноти часто використовують візуальне представлення даних у вигляді графіків і діаграм.

Середовище програмування PascalABC також може використовувати для побудови графіків та діаграм.

Завдання 1. Скласти програму побудови графіку функції y(x)=0.4x+2sin3x на проміжку [-15; 15].

Першим кроком визначаємо вхідні дані та задаємо розміри графічного вікна. Потім будуємо вісі координат та визначаємо координати точки графічного екрану, яка відповідає початку відліку побудованої системи координат x0=320, y0=200.

Далі визначаємо довжину одиничного відрізку в пікселях k=20.

Обчислення координат для побудови графіка організовуємо в циклі While.

Обчислюємо функцію y:=0.4\*x+2\*sin(3\*x), потім обчислюємо координати x1:=trunc(x0+x\*k); y1:=trunc(y0-y\*k). Функція trunc() відкидає дробову частину, оскільки координати точки повинні бути цілим числом. Для того, щоб забезпечити суцільну лінію графіка функції, збільшення аргументу для наступної точки повинно бути невеликим, наприклад: x=x+0,001.

Зазначимо, що перед циклом потрібно задати початкове значення аргументу функції, яке дорівнює лівій границі проміжку.

Програма може виглядати так:

Program Grafik;

uses GraphABC;

Var x0,y0, x1,y1, k : integer; x,y: real;

Begin

setWindowSize(640,400);

x0:=320; y0:=200; k:=20;

line(x0,20,x0,380); line(20,y0,620,y0); {осі координат}

Line(x0,20,x0-3,25); Line(x0,20,x0+3,25);

Line(620,y0,615,y0-3); Line(620,y0,615,y0+3);

TextOut(x0+5,20,'Y'); TextOut(615,y0+5,'X');

TextOut(10,10,'Графік функції y=0.4\*x+2\*sin(3\*x)');

for i:=0 to 29 do

line(20+k\*i,197, 20+k\*i,204); { позначки на осі абсцис }

for i:=1 to 20 do

line(317, 20+k\*i, 324, 20+k\*i); { позначки на осі ординат }

x:=-15; { початкове значення аргументу }

While x<=15 do

begin

y:=0.4\*x+2\*sin(3\*x); { обчислення значень функції }

x1:=trunc(x0+x\*k); y1:=trunc(y0-y\*k); { обчислення координат }

SetPixel(x1,y1,clRed);{ побудова }

x:=x+0.0005;

end;

End.

Приклад побудови діаграм розглянемо на наступному прикладі. Побудуємо кругову діаграму. Кругову діаграму використовують для наглядного зображення часток в загальному об'ємі даних.

Завдання 2. Є таблиця оцінок, отриманих учнями паралелі класів з математики. Скласти програму обчислення процентів учнів відповідно до їх досягнень.

Для початку показники успішності занесемо в цілочисленний масив A, а їх назви в рядковий масив B і масиви опишемо як постійні величини.

Кругова діаграма буде складатись з 4 секторів. Градусна міра кожного сектора визначається значенням відповідного елементу масиву.

Спочатку обчислюємо загальну кількість учнів, які отримали оцінки (це повне коло 360°). Площа кожної частини визначається за формулою – Оі\*360/S, де Оі – відповідна кількість оцінок певного рівня досягнень учнів, а S – загальна кількість учнів.

Для побудови кожного сектора потрібно знати значення двох кутів: початкового та кінцевого. Початковий кут першого сектору дорівнює 0°. Для обчислення кінцевого кута використовуємо наступну формулу: u0+round(Оі\*360/S), де u0 – початковий кут, а round() – функція, яка округлює значення до найближчого цілого. Цей кут буде початковим для наступного сектору. Колір сектору задаємо випадковим чином.

Програма може виглядати так:

Program Diagram2;

Uses crt, GraphAbc;

Const A: array[1..4] of integer = (5, 23, 22, 7);

B: array[1..4] of string = ('Початковий','Середній', 'Достатній', 'Високий');

Var u, u0, S, i: integer;

Begin

setWindowSize(400,400); s:=0; u0:=0;

for i:=1 to 4 do S:=S+A[i]; {обчислення загальної кількості учнів}

for i:=1 to 4 do

begin

Writeln(B[i],' рівень досягнень ',A[i],' учнів ', round(100\*A[i]/S), '%'); {виведення рядка таблиці}

SetBrushColor(rgb(random(255),random(255),random(255)));

u:=u0 + round(A[i]\*360/S); { обчислення кінцевого кута сектора}

Pie(220,240,140,u0,u); {малювання сектора }

u0:=u; {нове значення початкового кута для наступного сектора діаграми }

end;

End.

**Завдання**

1. Скласти програму побудови графіка функції:

а) y(x)=0.5xcos2x на проміжку [-12;12];

б) y(x)=8 sinxsin2x на проміжку [-15; 15].

2. Скласти програму, яка будує кругову діаграму, що ілюструє:

а) розподіл результатів голосування по п'яти кандидатам (дані вводяться з клавіатури);

б) хімічний склад земної кори (дані вводяться з клавіатури: кисень 47,2%, кремній 27,6%, алюміній 8,3%, залізо 5,1%, кальцій 3,6% , інші елементи 8,2 %).

## **Обробка подій в графічному вікні**

В мові програмування **Pascal ABC** для обробки подій в графічному вікні використовується спеціальний модуль **Events**.

Модуль **Events** призначений для створення програм, в яких використовують методи керування подіями. Даний модуль використовується в парі з модулем GraphABC. В ньому містяться команди для обробки подій, які відбуваються на пристроях введення (клавіатура та миша).

В випадку підключення модулів **GraphABC** і **Events** програма після запуску та виконання тіла програми не завершується, а продовжує виконуватися, відстежуючи події, що виникають в графічному вікні при використанні пристроїв введення. Тобто програма відстежує події, які стосуються пристроїв введення, а також дій, пов'язаних з графічним вікном (зміна розмірів та закриття вікна). Робота програми завершується тільки після події, яка закриває графічне вікно.

Кожній події відповідає відповідна процедурна змінна:

|  |  |
| --- | --- |
| Змінна | Подія |
| **OnMouseDown** | Натискання клавіші миші |
| **OnMouseUp** | Відпускання клавіші миші |
| **OnMouseMove** | Переміщення миші |
| **OnKeyDown** | Натискання будь-якої клавіші керування на клавіатурі |
| **OnKeyUp** | Відпускання клавіші клавіатури |
| **OnKeyPress** | Натискання символьної клавіші клавіатури |
| **OnResize** | Зміна розмірів графічного вікна |
| **OnClose** | Закриття графічного вікна |

До початку роботи програми значення процедурних змінних дорівнюють нулю. Для того щоб при настанні певної події була виконана певна дія, потрібно в програмі процедурній змінній присвоїти конкретну процедуру, яка буде виступати в ролі оброблювача подій. При виникненні однієї з вище перерахованих подій перевіряється чи містить відповідна процедурна змінна посилання на процедуру-оброблювач, якщо вона існує, то відповідна процедура-оброблювач виконується.

Процедурні змінні, які відповідають певним подіям, в модулі **Events** описані наступним чином:

**Var**

* OnMouseDown, OnMouseUp, OnMouseMove: **procedure** (x,y,mousebutton: integer); – параметри **х**, **у** в оброблювачах OnMouseDown, OnMouseUp, OnMouseMove визначають координату курсору миші в момент настання події, параметр **mousebutton** може приймати наступні значення: 0 – жодна клавіша миші не натиснута, 1 – натиснута ліва клавіша миші, 2 – натиснута права клавіша миші;
* OnKeyDown, OnKeyUp: **procedure** (key: integer); – параметр **key** в оброблювачахOnKeyDown, OnKeyUp визначає певний **віртуальний код** натиснутої клавіші;
* OnKeyPress: **procedure** (ch: char); – параметр **ch**в оброблювачіOnKeyPress визначає натиснутий символ, якщо змінна-подія OnKeyPress має оброблювач, то графічне вікно не закривається при натисканні клавіші Esc;
* OnResize, OnClose: **procedure**;.

Наприклад, якщо задана наступна процедура-оброблювач:

**procedure** MouseDown(x,y,mb: integer);

begin

write(1);

end;

яка в основній програмі зв'язана з відповідною процедурною змінною:

OnMouseDown:=MouseDown;

то всякий раз при натисканні клавіш миші в графічному вікні буде виводитись число 1.

Приклад такої програми:

Program p1;

uses GraphABC,Events,crt;

procedure MouseDown(x,y,mb: integer);

begin

write(1);

end;

begin

OnMouseDown:=MouseDown;

end.

**Віртуальні коди клавіш**

|  |  |  |  |
| --- | --- | --- | --- |
| **VK\_Left  VK\_Right  VK\_Up  VK\_Down  VK\_Back  VK\_Tab  VK\_Return  VK\_Escape  VK\_Space** | **VK\_Pause  VK\_CapsLock  VK\_NumLock  VK\_ScrollLock**  **VK\_Insert  VK\_Delete  VK\_Control  VK\_Shift  VK\_Alt** | **VK\_F1  VK\_F2  VK\_F3  VK\_F4  VK\_F5  VK\_F6  VK\_F7  VK\_F8  VK\_F9** | **VK\_F10  VK\_F11  VK\_F12**  **VK\_PageUp VK\_PageDown  VK\_Home  VK\_End** |

Алфавітно-цифрові клавіші мають віртуальний код, який дорівнює коду відповідного символу.

**Приклад програми для обробки подій.**

Програма, яка перемальовує різнокольорові прямокутники при зміні розмірів вікна мишкою.

Program pererisovka;

Uses graphabc, events;

var x,y,x1,y1,csh,cpl,w,h,a,b,i:integer;

procedure mousedown(x1,y1,mb:integer);

begin

end;

procedure mouseup (x1,y1,mb:integer);

begin

end;

procedure resize;

begin

ClearWindow;

w:=WindowWidth;

h:=WindowHeight;

a:=Trunc(w/30);

b:=20;

x:=Trunc((w-a\*30)/2);

y:=20;

For i:=0 to a-1 do

begin

rectangle(x,y,x+29,y+b);

floodfill(x+2,y+2,RGB(random(255),random(255),random(255)));

x:=x+30;

end;

end;

begin

ClearWindow;

w:=WindowWidth;

h:=WindowHeight;

writeln(w);

SetWindowCaption('Перерисовування');

a:=Trunc(w/30);

b:=20;

x:=Trunc((w-a\*30)/2);

y:=20;

For i:=0 to a-1 do

begin

rectangle(x,y,x+29,y+b);

floodfill(x+2,y+2,RGB(random(255),random(255),random(255)));

x:=x+30;

end;

onresize:=resize;

end.