Тема: Масиви.

Мета: Ознайомитися із однією із найбільш розповсюджених структур даних – масивами; навчитися вводити масиви, розраховувати для них базові характеристики, відшукувати елементи у масиві, сортувати елементи масиву.

1. Поняття масиву. Оголошення одновимірного масиву. Індексація елементів.

Масивом називається впорядкована індексована сукупність однотипних елементів, які мають спільне ім'я.

Елементами масиву можуть бути дані різних типів, у тому числі, й структурованих. Кожен елемент одновимірного масиву однозначно визначається іменем та індексом (для багатовимірних – індексами).

Масиви зручно використовувати там, де потрібно працювати із великою кількістю однотипних даних. При цьому передбачається багаторазова їх обробка.

Одновимірний (лінійний) масив – це масив, у якого в описі задано лише один індекс. Одновимірні масиви часто називають векторами, так як вони являють собою скінченну послідовність пронумерованих елементів.

Загальний вигляд опису лінійного (одновимірного) масиву:

Array [<тип індексів>] of <тип елементів>;

Частіше за все опис відбувається наступним чином:

Array [<нижня межа> .. <верхня межа>] of <тип елементів>;

Наприклад:

```
Var
  A: array [byte] of real;
  B: array [1..5] of integer;
  C: array [0..9] of boolean;
```

Зверніть увагу на те, що нижня та верхня межа індексів у масиві вказуються довільно. Границі діапазону можуть бути навіть від'ємними. Головне, щоб права межа діапазону була більшою від лівої.

На практиці, як правило, елементи нумеруються, розпочинаючи із 1 чи 0. Програмісту – початківцю також краще вести нумерацію, розпочинаючи із 1, так як при рахунку ми використовуємо саме таку послідовність значень. Проте, з точки зору збереження даних та деяких алгоритмів, більш доцільно розпочинати нумерацію з 0.

Нумерувати елементи масиву можна не тільки цілими числами. Довільний порядковий тип даних (перераховуваний, інтервальний, символьний, логічний, а також довільний, створений на їх основі) може виступати в ролі індексу. При цьому сукупний розмір масиву не може перевищувати 2 ГБ.

Таким чином, допустимі наступні описи масиву:

```
Var
  D: array [char] of real;
  E: array ['a' .. 'z'] of integer;
  F: array [Boolean] of char;
```

Приклади некоректних описів масиву:

```
  G: array [integer] of integer;
  H: array [real] of byte;
```

Масиви відносяться до структур прямого доступу. При цьому вміст масиву зберігається в неперервній області пам'яті. Це значить, що можна напряму (не перебираючи всі попередні) звернутися до потрібного елементу масиву. Для цього необхідно вказати ім'я масиву та індекс (індекси), розміщені в квадратних дужках:

```
a[1]:=5.4;
b[2]:=-2;
d['1']:=1.568;
a['b']:=7;
f[true]:='Y';
```

Подібний масив можна уявити як таблицю із одним рядком та декількома комірками:

Розміщення масиву А цілих чисел у пам'яті:

A[1]	A[2]	A[3]	A[4]	A[5]
Базова адреса	Базова адреса + 4 байти	Базова адреса + 8 байт	Базова адреса + 12 байт	Базова адреса + 16 байт

Такого типу масив, який являє собою набір однотипних даних, називається простим або одновимірним.

Всього під масив, наведений у таблиці, в пам'яті буде відведено 20 байт – оголошений масив А цілих чисел типу integer, кожному елементу (комірці) масиву буде виділено по 4 байти пам'яті. Всі ці елементи відповідають звичайним змінним цього типу.

2. Введення даних у масив. Відображення його вмісту.

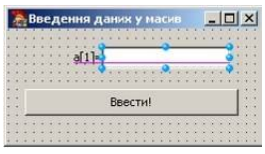
При вивченні мови програмування Паскаль Ви практикували введення даних до масиву, використовуючи нескладний цикл For:

```
var a: array [1..10] of integer;
      i: integer;
begin
 for i:=1 to 10 do
begin
  write ('a[', i, ']=');
  readln (a[i]);
end;
end;
```

Такий підхід у візуальному програмуванні не може бути застосований, оскільки повинна бути використана послідовність дій (введення даних, натискування кнопки) самого користувача.

Завдання 1. Розробити програму, яка реалізує заповнення на формі масиву цілих чисел, який містить 10 елементів (Massiv01).

Розробимо інтерфейс програми. Введення елементів масиву доцільно виконати за допомогою компонентів Label, Edit, Button:



Програмний код для реалізації завдання – наступний:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Edit1: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  a: array [1..10] of integer;
  i:integer=1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
 a[i]:=strtoint(edit1.text);
 label1.caption:='a[' + inttostr (i+1) + ']=';
 i:=i+1;
 edit1.clear;
 edit1.setfocus;
 if i=11 then
 begin
   label1.visible:=false;
   edit1.enabled:=false;
   button1.enabled:=false;
 end;

end;

end.
```

Деякі коментарі щодо рядків програмного коду:

```
 a[i]:=strtoint(edit1.text);
```

//запис значення із поля

```
 label1.caption:='a[' + inttostr (i+1) + ']=';
```

// відображення відповідного індексу чергового елементу масиву

```
 i:=i+1;
```

// збільшення поточного індексу

```
 edit1.clear;
 edit1.setfocus;
```

// очистити поле введення та перевести фокус

```
 if i=11 then
 begin
   label1.visible:=false;
   edit1.enabled:=false;
   button1.enabled:=false;
 end;
```

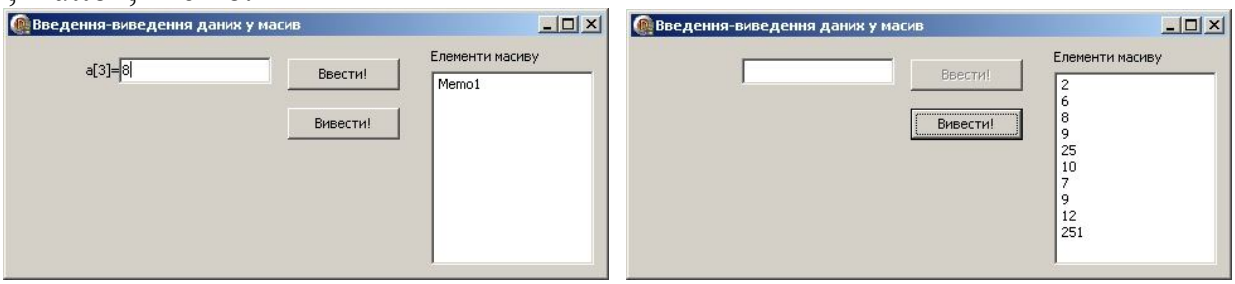
// якщо індекс стане рівним 11, то завершити введення

При вивченні мови програмування Паскаль Ви практикували виведення елементів масиву, використовуючи нескладний цикл For:

```
var a: array [1..10] of integer;
    i: integer;
begin
 for i:=1 to 10 do
        write ('a[', i, ']=', a[i]);
end;
```

Завдання 2. Доповнити попередній проект таким чином, щоб по завершенні введення елементи масиву виводились на форму (Massiv02).

Для організації виведення елементів масиву на формі розмістимо компоненти Label, Button, Memo.



Елементи виводяться у полі Memo по натискуванні на кнопку Вивести. Фрагмент програмного коду:

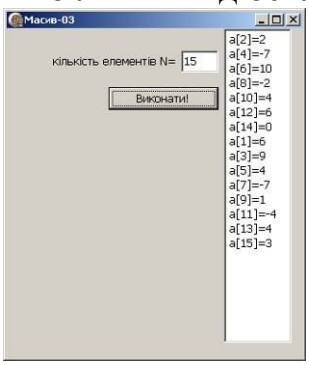
```
var
   Form1: TForm1;
   a: array [1..10] of integer;
   i:integer=1;
   k: integer;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
 a[i]:=strtoint(edit1.text);
 label1.caption:='a[' + inttostr (i+1) + ']=';
 i:=i+1;
 edit1.clear;
 edit1.setfocus;
 if i=11 then
 begin
    label1.visible:=false;
    edit1.enabled:=false;
    button1.enabled:=false;
 end;

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  for k:=1 to 10 do
    memo1.Lines.Add(inttostr(a[k]));
end;
```

Практичне завдання 1. Дано масив із N (N<=50) цілих чисел. Його потрібно заповнити випадковими числами у діапазоні від -10 до 10. Вивести спочатку всі елементи, розміщені на парних позиціях, а потім – всі елементи, розміщені на

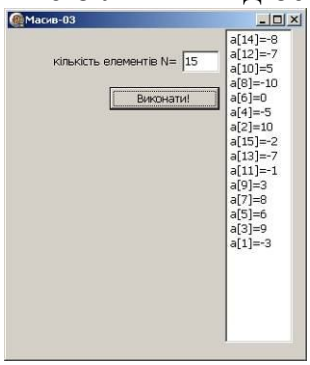
непарних позиціях. При цьому якщо перший елемент більший від останнього елементу, то виведення повинне здійснюватися від початку до кінця масиву, у протилежному випадку – від кінця до початку.

Розв'язок. При розв'язуванні цієї задачі потрібно звернути увагу на розмірність масиву. Завдання передбачає необхідність введення числа N (кількість елементів), при цьому його значення не може бути більшим від 50. Відповідно, нам цілком підійде для використання статичний масив (пам'ять для якого виділяється на етапі компіляції). У процесі роботи програми, можливо, будуть задіяні не всі елементи масиву. Це відбудеться, якщо користувач введе число, менше від 50. Роботу із статичним масивом демонструє наступна програма (Massiv03):

1 елемент більший від останнього



1 елемент не більший від останнього



Для реалізації завдання форма містить елементи Label (кількість елементів N=), Edit (для числа), Button (Виконати!), Memo (для виведення елементів масиву).

У розділі Var опишемо масив та дві змінні типу цілі числа:

```
var
  Form1: TForm1;
  a: array [1..50] of integer;
  i, n: integer;
```

У програмі доцільно використати дві процедури – відкриття форми та дія по натискуванні на кнопку Виконати.

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

У процедурі щодо відкриття форми доцільно прописати виконання процедури randomize – генератора випадкових чисел:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
end;
```

У процедурі обробника події «Натиснути на кнопку Виконати!» власне й реалізовано розв'язок задачі:

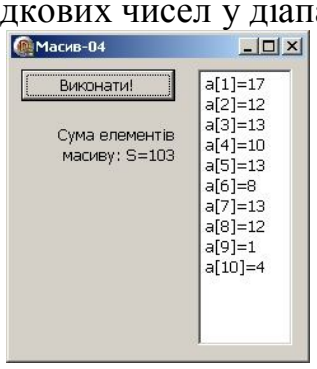
```
procedure TForm1.Button1Click(Sender: TObject);
begin
    memo1.Clear;
    n:=strtoint (edit1.Text); //ввести кількість елементів
    for i:=1 to n do
      a[i]:=random(21)-10;    //заповнити випадковими числами
    if a[1]>a[n] then         //якщо перший елемент більший від останнього,
    begin                     //вивести від 1 до останнього
      for i:=1 to n do
        if i mod 2 =0 then    //перевірка на парність
         memo1.lines.Add('a['+inttostr(i)+']='+inttostr(a[i]));
      for i:=1 to n do
        if i mod 2 <>0 then   //перевірка на непарність
         memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
    end
    else                      //якщо перший елемент не більший від останнього
    begin                     //виведення від останнього до першого
      for i:=n downto 1 do
        if i mod 2 = 0 then
         memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
      for i:=n downto 1 do
        if i mod 2 <> 0 then
         memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
    end;
end;
```

3. Визначення підсумкових показників для числового масиву.

Значна частина задач, які працюють із масивами чисел, передбачає визначення показників – суми, добутку чи середнього арифметичного елементів масиву.

Процес накопичення суми елементів масиву досить простий і практично нічим не відрізняється від додавання значень деякої числової послідовності. Змінній, у якій зберігається значення суми, присвоюється значення, рівне 0, потім у циклі послідовно додаються елементи масиву. Як правило, при роботі з масивом використовується цикл For – це пов’язане із тим, що в даній структурі кількість ітерацій наперед відома. Тип змінної для збереження суми повинен бути таким же, як і тип елементів масиву, або поглинати його (в деяких випадках розмір може виявитися недостатнім).

Завдання 3. Знайти суму усіх елементів масиву A, який складається із 10 цілих випадкових чисел у діапазоні від 0 до 20 (Massiv04).



Форма містить поле Memo, в якому при запуску програми на виконання відображаються десять елементів масиву, кнопку «Виконати!» та текстове поле «Сума елементів масиву».

У розділі Var опишемо масив та дві змінні типу цілі числа:

```
var
  Form1: TForm1;
  a: array [1..10] of integer;
  i, s: integer;
```

У програмі доцільно використати дві процедури – відкриття форми та дія по натискуванні на кнопку Виконати.

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

У процедурі щодо відкриття форми доцільно прописати виконання процедури randomize – генератора випадкових чисел, очищення вмісту поля Memo, генерацію елементів масиву та їх виведення на форму:

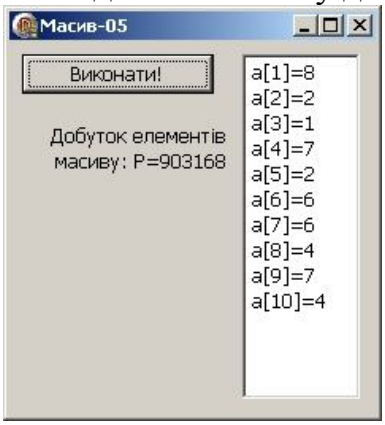
```
procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
 memo1.Clear;
 for i:=1 to 10 do                    //заповнити масив випадковими числами
   a[i]:=random(20);
 for i:= 1 to 10 do                   //вивести елементи масиву у полі Memo
   memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
end;
```

У процедурі обробника події «Натиснути на кнопку Виконати!» власне й реалізовано розв’язок задачі:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    s:=0;                  //початкова сума
    for i:=1 to 10 do      //додати всі елементи масиву
     s:=s+a[i];
    label1.Caption:='Сума елементів'+#13+'масиву: S='+inttostr(s);
                           //вивести значення суми
end;
```

Пошук добутку елементів масиву виконується аналогічним чином. Відмінність полягає лише у тому, що початкове значення добутку повинне бути рівним 1.

Завдання 4. Знайти добуток усіх елементів масиву A, який складається із 10 цілих випадкових чисел у діапазоні від 1 до 10 (Massiv05).

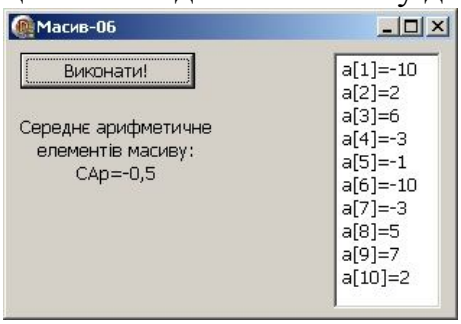


Програмний код для розв’язування цієї задачі наступний:

```
var
  Form1: TForm1;
  a: array [1..10] of integer;
  i, p: integer;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    p:=1;                 //початковий добуток
    for i:=1 to 10 do     //перемножити всі елементи масиву
     p:=p*a[i];
    label1.Caption:='Добуток елементів'+#13+'масиву: P='+inttostr(p);
                          //вивести значення добутку
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
 memo1.Clear;
 for i:=1 to 10 do                //заповнити масив випадковими числами
   a[i]:=random(10)+1;
 for i:= 1 to 10 do               //вивести елементи масиву у полі Memo
   memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
end;

end.
```

Для пошуку середнього арифметичного значення всіх елементів масиву необхідно визначити суму, а потім розділити знайдене значення на кількість елементів. Зверніть увагу на те, що ділення відбувається за межами циклу, а значення змінної середнього арифметичного повинне бути дійсного типу.

Завдання 5. Знайти середнє арифметичне елементів масиву А, який складається із 10 цілих випадкових чисел у діапазоні від -10 до 10 (Massiv06).

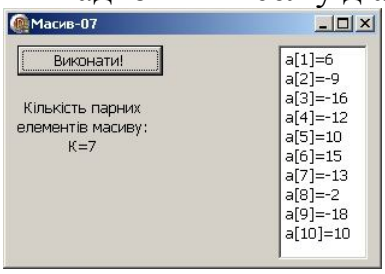


Програмний код для розв’язування цієї задачі наступний:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    s:=0;                 //початкова сума
    for i:=1 to 10 do     //додати всі елементи масиву
     s:=s+a[i];
    sar:=s/10;            //середнє арифметичне
    label1.Caption:='Середнє арифметичне'+#13+
                    'елементів масиву:'+#13+ 'CAp='+floattostr(sar);
                          //вивести значення середнього арифметичного
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
 memo1.Clear;
 for i:=1 to 10 do                //заповнити масив випадковими числами
   a[i]:=random(20)-10;
 for i:= 1 to 10 do               //вивести елементи масиву у полі Memo
   memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
end;
```

У задачах із використанням масивів досить часто виникає необхідність знайти кількість елементів, які задовольняють певним критеріям. Умови обмеження можуть бути різноманітними – наприклад, підрахунок кількості додатних, від'ємних, парних, непарних тощо елементів.

Завдання 6. Знайти кількість парних елементів масиву А, який складається із 10 цілих випадкових чисел у діапазоні від -20 до 20 (Massiv07).



Програмний код для розв'язування цієї задачі наступний:

```
var
  Form1: TForm1;
  a: array [1..10] of integer;
  i, k: integer;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    k:=0;                  //початкова кількість
    for i:=1 to 10 do      //перевірити всі елементи масиву
     if (a[i] mod 2 =0) then k:=k+1;     //якщо елемент парний, то
                           //кількість збільшити на 1
    label1.Caption:='Кількість парних'+#13+
                    'елементів масиву:'+#13+ 'K='+inttostr(k);
                           //вивести значення кількості
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
 memo1.Clear;
 for i:=1 to 10 do                 //заповнити масив випадковими числами
   a[i]:=random(40)-20;
 for i:= 1 to 10 do                //вивести елементи масиву у полі Memo
   memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
end;

end.
```

4. Пошук даних у масиві.

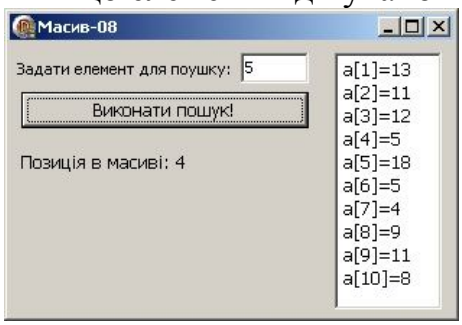
Пошук – одна із дій, яка найчастіше зустрічається у задачах по програмуванню. Вхідними даними для реалізованого алгоритму є послідовність елементів a[1], a[2], a[3], …, a[N] (у нашому випадку це масив) та деякий елемент x. Задача полягає у тому, щоб отримати на виході номер елементу масиву, рівного x, або з'ясувати, що такого елементу в послідовності немає.

Завдання 7. Скласти програму для реалізації послідовного пошуку елементу у масиві (Massiv08).

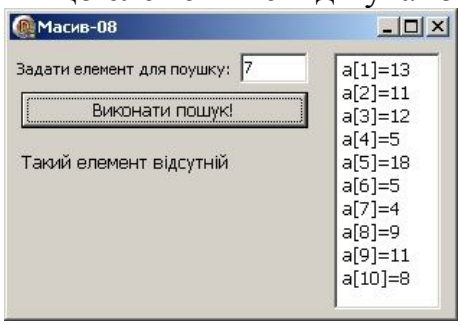
Розв'язок. У відповідності із методом послідовного (лінійного) пошуку почергово порівнюються елементи масиву із заданими значенням. Виявивши такий елемент, повертаємо його індекс. Якщо по завершенні порівняння вказаний елемент не знайдено, то зберігаємо інформацію про це.

Вигляд форми проекту, який реалізовує послідовний пошук

Якщо елемент відшукано



Якщо елемент не відшукано



Програмний код проекту:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    x:=strtoint (labeledit1.Text); //ввести елемент для пошуку
    p:=-1;   //початкове значення позиції
    for i:=1 to 10 do     //перевірити всі елементи масиву
     if (a[i] = x) then  //якщо знайшли елемент
     begin
        p:=i;            //зберігаємо позицію
        break;           //завершуємо пошук
     end;
     if p<>-1 then       //якщо елемент відшукано
       label1.Caption:='Позиція в масиві: '+inttostr(p)
     else                //якщо елемент відсутній
       label1.Caption:='Такий елемент відсутній';
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
 memo1.Clear;
 for i:=1 to 10 do                //заповнити масив випадковими числами
   a[i]:=random(20);
 for i:= 1 to 10 do               //вивести елементи масиву у полі Memo
   memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
end;

end.
```

26: 22 Insert Code Design History

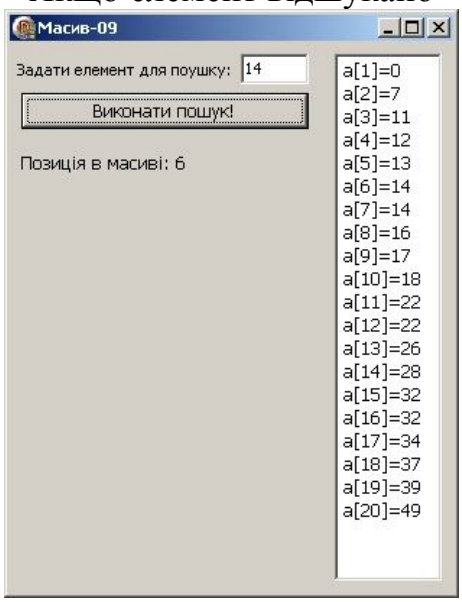
Запропонований алгоритм має один суттєвий недолік – у процесі пошуку доводиться перевіряти в середньому половину масиву (при наявності шуканого елементу у масиві) або ж весь масив (у випадку відсутності елементу). При великих розмірах масиву це дуже неекономно.

Завдання 8. Скласти програму для реалізації бінарного способу пошуку елементу у масиві (Massiv09).

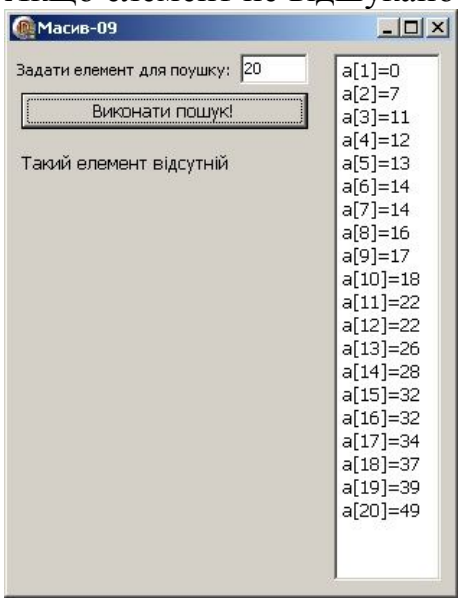
Розв'язок. Кількість операцій можна суттєво зменшити за допомогою бінарного (двійкового) пошуку. Це класичний алгоритм пошуку елементу у відсортованому масиві. Суть алгоритму полягає у зменшенні (в 2 рази) на кожному кроці області для пошуку.

Розглянемо послідовність дій для даного виду пошуку. Беремо середній елемент – якщо він дорівнює шуканому, то пошук успішно завершується. Якщо середній елемент більше шуканого, то шуканий елемент знаходиться у першій половині масиву (якщо він є у масиві). Якщо ж середній елемент менше шуканого, то шуканий елемент знаходиться у другій половині масиву. Таким чином, ми розбили масив на дві частини. Використовуємо той же алгоритм у вибраній частині масиву. Продовжуємо дії до тих пір, поки не буде знайдено шуканий елемент або доведено, що він у масиві відсутній.

Якщо елемент відшукано



Якщо елемент не відшукано



Програмний код проекту:

У процедурі, що описує подію OnCreate форми, окрім формування масиву із випадкових елементів також забезпечено й сортування елементів масиву за зростанням.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 randomize;
 memo1.Clear;
 for i:=1 to 20 do                  //заповнити масив випадковими числами
   a[i]:=random(50);
 for j:=1 to 20 do                  //відсортувати масив за зростанням
    begin
    for I := 2 to 20 do
      begin
        if a[i]<a[i-1] then
        begin
          middle:=a[i];
          a[i]:=a[i-1];
          a[i-1]:=middle;
        end;
      end;
    end;
 for i:= 1 to 20 do                 //вивести елементи масиву у полі Memo
   memo1.lines.add('a['+inttostr(i)+']='+inttostr(a[i]));
end;

end.
```

У процедурі, що описує подію OnClick кнопки «Виконати пошук!» власне й реалізовано бінарний алгоритм пошуку елементу:

```
var
  Form1: TForm1;
  a: array [1..20] of integer;
  i,j,l,x,p,middle: integer;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
    x:=strtoint (labeledit1.Text); //ввести елемент для пошуку
    l:=1;   //початкове значення лівої позиції
    p:=20;  //початкове значення правої позиції
    while l<=p do
    begin
      middle:=round((p+l)/2);  //обрахувати середину
      if x=a[middle] then      //порівняти шукане із середнім
      begin
        label1.Caption:='Позиція в масиві: '+inttostr(middle);
        exit;
      end
      else
       if x<a[middle] then     //шуканий елемент у лівій частині
        p:=middle-1            //перенести праву границю
       else                    //шуканий елемент у правій частині
        l:=middle+1;           //перенести праву границю
    end;
   label1.Caption:='Такий елемент відсутній';
end;
```

Важливо! У наведеній програмі, яка реалізовує алгоритм бінарного пошуку, елементи повинні бути відсортовані.

5. Теоретичні питання для самоконтролю.

1. Наведіть приклади задач, які можуть бути розв’язані за допомогою одновимірних масивів.
2. Дайте опис одновимірного масиву у загальному вигляді та на конкретних прикладах.
3. Які типи даних можуть бути використані в ролі індексів масиву? Наведіть декілька прикладів.
4. Яка конструкція використовується для введення масиву?
5. Чим відрізняється послідовність дій при пошуку суми від добутку елементів масиву?
6. Які типи алгоритмів, що реалізовують пошук елемента у масиві, Ви знаєте? Який із них, на Вашу думку, більш ефективний?