**program exercice\_2;**

uses wincrt;

var u,u1,u2,x :integer;

**Exercice 2:**

Soit la suite (U) définie par :

U0 = 2

U1 = 3

Un = Un-1 + 2 \* Un-2 ; pour tout n>=2

En supposant que cette suite est croissante, écrire un programme pascal permettant de lire un entier x (x>2), de vérifier et d'afficher s'il est un terme de la suite U ou non. Dans l'affirmative afficher son rang.

**procedure lectx(var x :integer);**

begin

repeat

readln(x);

until x>2 ;

end;

**function recherchex( x: integer) :boolean;**

var

u0, u1, u :integer;

trouve : boolean;

begin

u0 := 2;

u1 := 3;

trouve := false;

repeat

u:= u1 + 2 \* u0 ;

if u = x then trouve :=true;

u0 := u1;

u1 := u ;

until u >= x;

**recherchex := trouve;**

end;

**function rang( x: integer) :integer;**

var

u0, u1, u, i :integer;

begin

u0 := 2;

u1 := 3;

i := 1;

repeat

u:= u1 + 2\*u0;

i:=i+1;

u0 := u1;

u1 := u ;

until u = x;

**rang :=i;**

end;

**begin**

lectx(x);

**if** x=3 then write(x, ' est un terme de suite U, de rang : 1')

**else**

**if** recherchex(x)= false **then** write(x,' n est pas un terme de la suite U')

**else** write(x, ' est un terme de suite U, de rang : ',rang(x));

**end.**

**Exercice 1: Mise à zéro de la diagonale principale d'une matrice**

Ecrire un programme qui met à zéro les éléments de la diagonale principale d'une matrice ***carrée*** Adonnée.

**Exercice 2: Matrice unitaire**

Ecrire un programme qui construit et affiche une matrice ***carrée unitaire*** U de dimension N. Une matrice unitaire est une matrice, telle que:



**Problème 1: (Carré magic)**

Un carré magique est un arrangement de nombres avec **n** lignes et **n** colonnes tel que la somme des valeurs de chaque ligne = la somme des valeurs de chaque colonne = la somme des valeurs de chaque diagonale.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 16 | 9 | 2 | 7 |
| 2 | 6 | 3 | 12 | 13 |
| 3 | 11 | 14 | 5 | 4 |
| 4 | 1 | 8 | 15 | 10 |

Par exemple, le carré suivant est magique

La somme des valeurs de chaque ligne = 34

La somme des valeurs de chaque colonne = 34

La somme des valeurs de chaque diagonale = 34  
**Partie I :**

1) Ecrire une analyse et un algorithme d’une fonction intitulée **Somme\_ligne** qui permet de retourner la somme d’une ligne donnée.

2) Ecrire une analyse et un algorithme d’une fonction intitulée **Somme\_colonnne** qui permet de retourner la somme d’une colonne donnée.

3) Ecrire une analyse et un algorithme d’une fonction intitulée **Somme\_ diagonale1** qui permet de retourner la somme du 1ere diagonale.

4) Ecrire une analyse et un algorithme d’une fonction intitulée **Somme\_diagonale2** qui permet de retourner la somme du 2ème diagonale.

5) Ecrire une analyse et un algorithme d’une fonction intitulée **Magic\_1** qui permet de vérifier si un carré donné est magique ou non.

***Exercice***

La racine carrée approchée d’un nombre réel **R** par la méthode de Newton, définit de la façon suivante : **U0 = R**  **Un+1 = (Un+ R / Un) / 2**

**Cette suite converge** vers le calcul est arrêté lorsque | R - Un2| < e, où **e** est un réel positif saisi au clavier.

|  |  |
| --- | --- |
| uses wincrt;  var  e:real;  R:real;  Procedure saisie\_r(var R :real);  begin  repeat  Write('Donner un real > 0 :');  readln(R);  until R > 0;  end;  Procedure saisie\_e(var e : real);  begin  repeat  Write('Donner e dans[0..0,1] :');  readln(e);  until (e>0) and (e <=0.1);  end;  function racine:real;  var  U0,U:real;  begin  U0:=R;  Repeat  U:=(U0 + R/U0)/2;  U0:=U;  until abs(R-sqr(U)) < e;  racine:=U;  end;  begin  Saisie\_r(r);  saisie\_e(e);  writeln(racine);  end. |  |