Le 17/01/11
Durée : 1h30

EXAMEN DU MODULE ALGO

Exercice 1 (6 pts)

- Ecrire une fonction ***Puissance*** qui calcul x^n , ***x*** un réel et ***n*** un entier positif.

On peut représenter un polynôme $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$

à coefficient réels $a_0,\ldots,a_n$ et de degré ***n***, par un tableau ***T*** tel que $T[i]=a_i$

a) Ecrire une fonction ***LirePoly*** qui étant donné ***n,*** lit les coefficients et les sauvegarde dans ***T***.

b) A l'aide de la fonction ***Puissance*** précédente, écrire une fonction ***EvaluePoly*** qui évalue un polynôme ***P*** en un point ***x*** donnée de type réel. Cette fonction retourne la valeur de ***P(x)***.

c) Si l'on suit le déroulement de l'exécution de la fonction ***EvaluePoly***, on observe qu'elle fait beaucoup de multiplications inutiles (quand on calcul a^{i+1}, on recalcule a^i). Un moyen d'éviter cela est la fonction de *Horner*. Elle consiste à remarquer qu'on peut écrire $P(x) = (((\ldots(a_n x + a_{n-1})\, x + \ldots)x + a_2)x + a_1)x + a_0$

Ecrire une fonction ***Horner*** qui évalue le polynôme P.

Exercice 2 (10 pts) : Le problème suivant concerne un réseau de stations de bus.

Partie I :

a. Définir le type ***station*** qui contient le numéro et le nom (taille max=30) de la station.
b. Ecrire la fonction **station creerStation( )** qui initialise une station.
c. Ecrire la fonction **int memeStation(station a, station b)** qui retourne 1 si les stations a et b sont les mêmes (même nom et même numéro)

Partie II :

Deux stations sont reliées par un chemin (ou arc). Un arc contient la ***station de départ*** et la ***station d'arrivée.***

a. Définir le type ***arc***.
b. Ecrire la fonction ***arc creerArc()*** qui initialise un arc reliant deux stations ***a*** et ***b***.
c. Ecrire la fonction ***int appartient(station a, arc c)*** qui retourne 1 si la station ***a*** appartient à l'arc ***c***, 0 sinon.
d. Ecrire la fonction ***int memeArc(arc c1, arc c2)*** qui retourne 1 si les arcs ***c1*** et ***c2*** sont les mêmes.
e. Ecrire la fonction ***int seSuivent(arc c1, arc c2)*** qui retourne 1 si les arcs ***c1*** et ***c2*** se suivent (c'est-à-dire sont différents et sont reliés par une même station).

Partie III :

Un réseau de stations de bus est défini par un ensemble d'arcs reliant un ensemble de stations. Un réseau est une liste chainée d'arcs.

a. Déclarer la liste chainée
b. Ecrire une fonction ***créerReseau*** qui crée une liste ***L*** avec un nombre d'arc **nbr** donnée.
c. Ecrire une fonction ***rechercheStation*** qui étant données une station ***ST*** retourne son adresse (l'adresse de l'arc contenant cette station) si la station existe dans la liste ***L*** sinon retourne NULL.
d. Ecrire une fonction ***rechercheArc*** qui étant données une station de départ ***SD*** et une station d'arrivée ***SA*** retourne son adresse si l'arc existe dans la liste ***L*** sinon retourne ***NULL.***
e. Ecrire une fonction ***supprimArc*** qui permet de supprimer un arc ***C*** donnée de la liste ***L***, s'il existe.
f. La liste ***L*** des arcs est triée par ordre croissant des stations de départ et d'arrivé. Ecrire une fonction ***insertArc*** qui permet d'insérer un arc ***C*** donnée dans la liste ***L***.
g. Ecrire la fonction ***main*** qui crée un réseau de stations de bus (création de la liste d'arcs) puis affiche toutes les stations d'arrivé d'une station ***SD*** donnée (on suppose qu'un seul bus part de ***SD***.

Exercice 3 (4 pts)

Soient deux piles ***p1*** et ***p2*** d'entiers triés dans l'ordre croissant (le maximum au sommet). Ecrire une fonction qui crée une pile ***p3*** qui va contenir les valeurs de ***p1*** qui n'existent pas dans ***p2*** et les valeurs de ***p2*** qui n'existent pas dans ***p1***. (La pile ***p3*** obtenu est aussi triée dans l'ordre croissant).

NB :

Soient deux chaînes de caractères ch1 et ch2, on donne les fonctions :

strcmp (ch1, ch2) : retourne 0 si les chaînes sont égales.

strcpy (ch1, ch2) : copie la chaîne ch2 dans ch1.

On suppose que les primitives de manipulation des piles sont prédéfinies.

Bon courage

Corrigé de l'examen

Exercice1 :

```
float Puissance(float x, int n)
{ int i ; float p=1 ;
  for (i=1 ;i<n ;i++) p=p*x;
  return p;
}
```

a)
```
void LirePoly(float *T, int n)
{ int i;
  for (i=0;i<=n;i++)
    scanf("%f",&T[i]);
}
```

b)
```
float EvaluePoly(float *T, int n , float x)
{ float poly=0; int i;
  for(i=0;i<=n;i++)
    poly=poly+T[i]*Puissance(x,i) ;
return poly ;
}
```

c)
```
float Horner(float *T,int n, float x)
{ float poly=T[n]; int i;
  for(i=n-1;i>=0;i--)
    poly=poly*x+T[i];
return poly;
}
```

Exercice 2:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#include<stdlib.h>

/********************* Partie I ************************/
typedef struct {int num;
                char nom[30]; } station;
```

```
station  creerStation()
{ station  S;
  printf("donnez le numero de la station ");
  scanf("%d",&(S.num));
  printf("\ndonnez le nom de la station");
  scanf("%s",S.nom);
  return S;
}

int memeStation(station a, station b)
{ if(a.num==b.num && strcmp(a.nom,b.nom)==0)  return 1;
  return 0;
}

/******************** Partie II ************************/
typedef struct {station D, A; } arc;

arc creerArc()
{ arc C;
  C.D=creerStation();
  C.A=creerStation();
  return C;
}

int appartient(station ST, arc C)
{ if (memeStation(ST,C.D) || memeStation(ST,C.A)) return 1;
  return 0;
}

int memeArc(arc C1, arc C2)
{ if (memeStation(C1.D,C2.D) && memeStation(C1.A, C2.A)) return 1;
  return 0;
}

int seSuivent(arc C1, arc C2)
{ if(memeStation(C1.A,C2.D) || memeStation(C2.A,C1.D)) return 1;
  return 0;
}
```

```
/****************************** Partie III ****************************/
typedef struct no *liste;
typedef struct no { arc elt;
                    liste svt; } noeud;
liste creernoeud()
{ liste tete=(liste)malloc(sizeof(noeud));
  if (!tete) {printf("erreur");exit(-1); }
  return(tete);
}

liste creerListe(int nbr)
{ liste tete, p=NULL; int i;
  for(i=1;i<=nbr;i++)
    {  tete=creernoeud();   tete->elt=creerArc();    tete->svt=p;
       p=tete;
    }
  return tete;
}

liste rechStation(liste tete,station ST)
{ liste p=tete;
   while(p!=NULL && appartient(ST,p->elt.D)==0) p=p->svt;
  return p;
}

liste rechArc(liste tete, station SD, station SA)
{liste p=tete;  arc C; C.D=SD; C.A=SA;
   while(p!=NULL && memeArc(C,p->elt)==0) p=p->svt;
   return p;
}

void supprim(liste *tete, arc C)
{ liste p=*tete,prd;
  while(p!=NULL && memeArc(C,p->elt)==0)
   {prd=p; p=p->svt; }
 if (p!=NULL)
  { if (p==*tete) *tete=(*tete)->svt;
    else prd->svt=p->svt;
    free(p);
  }
}
```

```
/********************************** ou bien  **********************/
void supprim2(liste *tete,arc C)
{ liste p, prd;
  p=rechArc(*tete,C.D,C.A);
  if (p!=NULL)
    { prd=*tete;
      while(prd->svt!=p) prd=prd->svt;
      if (p==*tete) *tete=(*tete)->svt;
      else prd->svt=p->svt;
      free(p);
    }
}
/***********************************************/
void insert(liste *tete, arc C)
{ liste p=*tete, prd, r=creernoeud(); r->elt=creerArc();
   while(p!=NULL && p->elt.D.num<C.D.num)
    {prd=p;p=p->svt;}
   while(p!=NULL && p->elt.D.num==C.D.num && p->elt.A.num<C.A.num)
    {prd=p;p=p->svt; }
   if(p==*tete) {r->svt=*tete; *tete=r;}
   else { r->svt=p; prd->svt=r; }
}

void afficheStation(station S)
{ printf(" num = %d , nom = %s",S.num,S.nom);
}

main()
{ liste tete,p,R; int nbr; station SD;
  printf("Donnez le nombre d'arcs à creer\n");  scanf("%d",&nbr);
  tete=creerListe(nbr);
  printf("donnez la station de depart à rechercher \n");  SD=creerStation();
  p=tete;
  while(p!=NULL)
  { R=rechStation(p,SD);
    if (R!=NULL) { afficheStation(R->elt.A);
                   p=R->svt; }
    else p=NULL;
  }
  getch();
}
```

Exercice 3 :

```
typedef   int  typelem ;

pile   Difference_Symetrique(pile p1, pile p2)
{ pile p3=initPile(); typelem x;
  while ( !pilevide(p1) && !pilevide(p2))
  { if(sommetPile(p1)>sommetPile(p2))
       { desempiler(&p1,&x); empiler(&p3,x);  }
   else if(sommetpile(p1)<sommetPile(p2))
            {desempiler(&p2,&x);empiler(&p3,x); }
          else {desempiler(&p1,&x) ;  desempiler(&p2,&x);  }
  }
   if ( !pilevide(p1)) copiePile(p3,p1) ;
   else if (!pilevide(p2)) copiePile(p3,p2) ;
}
```