Un livre de Wikilivres.

Exercices en langage C++

Avant propos

Avant propos

Nous proposons ici de recueillir tous types d'exercices en langage C++, du plus basique au plus complexe. N'hésitez pas à rajouter ici et là un nouvel exercice avec sa solution.

Réaliser un programme qui utilise un menu pouvant effectuer les opérations suivantes :

1. Fusion de deux vecteurs pour en faire un seul à la fin
2. Produit d'une matrice par trace matricielle
3. Intersection des deux vecteurs
4. Quitter le programme.

Le programme doit permettre aussi a l'utilisateur de retourner dans le menu à chaque instruction.

N.B-Type matrice [4][4]

Notions de base

Notions de base

EXERCICE 1

Écrire un programme qui affiche ceci à l'écran :

```
Hello world!
Voici un programme illustrant l'utilisation de cout !
```

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- L'utilisation de cout et de `endl`.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main(int argc, char **argv)
{
  cout << "Hello world!" << endl;
  cout << "Voici un programme illustrant l'utilisation de cout !";
  cout << "Appuyez sur une touche pour continuer ..." << endl;
  cin.ignore();
  cin.get();
  return EXIT_SUCCESS;
}
```

EXERCICE 2

Écrire un programme qui demande à l'utilisateur de taper la largeur et la longueur d'un champ et qui en affiche le périmètre et la surface.

Solution

```
#include<iostream>
using namespace std;
int main()
{
double largeur,longueur,surface, perimetre;

cout << "Tapez la largeur du champ : "; cin >> largeur;
cout << "Tapez la longueur du champ : "; cin >> longueur;

surface = largeur * longueur;
perimetre = 2 * (largeur + longueur);
```

```
cout << "La surface vaut : " << surface << endl;
cout << "Le perimetre vaut : " << perimetre << endl;

cout << "Appuyez sur une touche pour continuer." << endl;
cin.ignore();
cin.get();

return EXIT_SUCCESS;
}
```

EXERCICE 3

Écrire un programme qui demande à l'utilisateur de taper 5 entiers et qui affiche leur moyenne. Le programme ne devra utiliser que 2 variables.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La notion de variables et leur déclaration.
- Calcul du moyenne.
- Utilisation des types int et double.
- L'utilisation de cin et de cout.
- L'affectation.

Voici le fichier source :

```
#include<iostream>
using namespace std;
int main()
{
int a;double s=0;

cout<<"Tapez la valeur numero 1 : ";cin>>a;s=s+a;
cout<<"Tapez la valeur numero 2 : ";cin>>a;s=s+a;
cout<<"Tapez la valeur numero 3 : ";cin>>a;s=s+a;
cout<<"Tapez la valeur numero 4 : ";cin>>a;s=s+a;
cout<<"Tapez la valeur numero 5 : ";cin>>a;s=s+a;

s=s/5.0;
cout<<"La moyenne vaut : "<<s<<endl;

cout << "Appuyez sur une touche pour continuer ..." << endl;
cin.ignore();
cin.get();

return EXIT_SUCCESS;
}
```

EXERCICE 4

Écrire un programme qui demande à l'utilisateur de saisir 2 entiers A et B, qui échange le contenu des variables A et B puis qui affiche A et B.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La notion de variables et leur déclaration.
- L'utilisation de cin et de cout.
- L'affectation.
- Un "algorithme" rudimentaire : échanger le contenu de 2 variables.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int a,b,temp;

cout<<"Tapez la valeur de a : ";cin>>a;
cout<<"Tapez la valeur de b : ";cin>>b;

temp=a;
a=b;
b=temp;

cout<<"La valeur de a est "<<a<<endl;
cout<<"La valeur de b est "<<b<<endl;

cout << "Appuyez sur une touche pour continuer ..." << endl;
cin.ignore();
cin.get();

return EXIT_SUCCESS;
}
```

EXERCICE 5

Écrire un programme qui demande à l'utilisateur de taper le prix HT d'un kilo de tomates, le nombre de kilos de tomates achetés, le taux de TVA (Exemple 10%,20%,...). Le programme affiche alors le prix TTC des marchandises.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La notion de variables et leur déclaration.
- Le choix d'identificateurs pertinents et explicites.
- L'utilisation de cin et de cout.
- L'affectation.
- Modélisation d'un problème "économique".

Voici le fichier source :

```
#include<iostream>
using namespace std;
int main()
{
double prixht,poids,tva,total;
```

```
cout<<"Tapez le prix HT d'un kilo de tomates : ";cin>>prixht;
cout<<"Combien de kilos avez-vous achetes : ";cin>>poids;
cout<<"Quel est le taux de TVA : ";cin>>tva;

total=(1+tva/100)*prixht*poids;

cout<<"Le prix TTC est : "<<total<<endl;

cout << "Appuyez sur une touche pour continuer ..." << endl;
cin.ignore();
cin.get();

return EXIT_SUCCESS;;
}
```

Structures de contrôle

Structures de contrôle

EXERCICE 1

Écrire un programme qui demande à l'utilisateur de taper un entier et qui affiche GAGNE si l'entier est entre 56 et 78 bornes incluses PERDU sinon.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La notion de variables et leur déclaration.
- L'utilisation de cin et de cout.
- Le choix d'une structure de contrôle adaptée au problème !

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{

int a;
cout<<"Tapez un entier : ";cin>>a;
if((a>=56)&&(a<=78))cout<<"GAGNE"<<endl; else cout<<"PERDU"<<endl;

cout << "Appuyez sur une touche pour continuer ..." << endl;
cin.ignore();
cin.get();
return EXIT_SUCCESS;
}
```

EXERCICE 2

Ecrire un programme qui affiche tous les entiers de 8 jusqu'à 23 (bornes incluses) en utilisant un for.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation rudimentaire d'un for.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
```

```
int i;
for(i=8;i<=23;i++)cout<<i<<endl;

return 0;
}
```

EXERCICE 3

Même exercice mais en utilisant un while.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation rudimentaire d'un while.

Voici le fichier source :

```
#include<iostream>
using namespace std;
int main()
{

int i=8;
while(i<=23)
    {
    cout<<i<<endl;
    i++;
    }

return 0;
}
```

EXERCICE 4

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers et qui affiche leur somme.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation d'un for.
- Etude d'un algorithme usuel : calcul d'une somme.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int i,s=0,x;
```

```
for(i=0;i<10;i++)
    {
    cout<<"Tapez un entier : ";cin>>x;
    s=s+x;
    }

cout<<"La somme vaut : "<<s<<endl;

return 0;
}
```

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers et qui affiche le plus petit de ces entiers.

Solution

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{

int i,ppt,x;

for(i=0;i<10;i++)
    {
    cout<<"Tapez un entier : ";cin>>x;
    if(i==0)ppt=x;else if(x<ppt)ppt=x;
    }

cout<<"Le plus petit vaut vaut : "<<ppt<<endl;

return 0;
}
```

EXERCICE 6

Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule la somme des cubes de 5^3 à N^3.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple du for.
- Etude d'un algorithme usuel : calcul d'une somme.
- Modélisation d'un problème simple issu des mathématiques.

Voici le fichier source :

```
#include<iostream.h>
```

```
using namespace std;
int main()

{
    int N;
    double somme;
    cout << "Entrer un entier : "; cin >> N;
    if (N>=5) {
        for (int i=5; i<=N; i++) somme += i*i*i;
    }
    cout << "Somme des cubes de 5^3 a " << N << "^3 = " << somme << endl;
    cin.ignore();
    cin.get();
    return EXIT_SUCCESS;
}
```

EXERCICE 7

Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule u(N) défini par :
u(0)=3
u(n+1)=3.u(n)+4

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple du for.
- Etude d'un algorithme usuel : calcul des termes d'une suite récurrente.
- Modélisation d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace;
int main()
{
int i,u=3,N;

cout<<"Tapez N : ";cin>>N;

for(i=0;i<N;i++)
    u=u*3+4;

cout<<"u("<<N<<")="<<u<<endl;

return 0;
}
```

EXERCICE 8

Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule u(N) défini par :
u(0)=1
u(1)=1
u(n+1)=u(n)+u(n-1)

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple du for.
- Etude d'un algorithme usuel : calcul d'une suite récurrente.
- Modélisation d'un problème simple issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int i,u=1,v=1,w,N;

cout<<"Tapez N : ";cin>>N;

w=1;

for(i=2;i<=N;i++)
    {
    w=u+v;
    u=v;
    v=w;
    }

cout<<"u("<<N<<")="<<w<<endl;

return 0;
}
```

EXERCICE 9

Ecrire un programme qui demande à l'utilisateur de taper un entier N entre 0 et 20 bornes incluses et qui affiche N+17. Si on tape une valeur erronée, il faut afficher "erreur" et demander de saisir à nouveau l'entier.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple du while.
- Validation des données saisies par l'utilisateur.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int N;
bool ok;

do
    {
```

```
    cout<<"Tapez N entre 0 et 20 :";cin>>N;
    ok= N<=20 && N>=0;
    if(!ok)cout<<"ERREUR RECOMMENCEZ"<<endl;
    }while(!ok);

N=N+17;
cout<<"La valeur finale est : "<<N<<endl;

return 0;
}
```

EXERCICE 10

Ecrire un programme qui permet de faire des opérations sur un entier (valeur initiale à 0). Le programme affiche la valeur de l'entier puis affiche le menu suivant :
1. Ajouter 1
2. Multiplier par 2
3. Soustraire 4
4. Quitter

Le programme demande alors de taper un entier entre 1 et 4. Si l'utilisateur tape une valeur entre 1 et 3, on effectue l'opération, on affiche la nouvelle valeur de l'entier puis on réaffiche le menu et ainsi de suite jusqu'à ce qu'on tape 4. Lorsqu'on tape 4, le programme se termine.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation du while.
- Utilisation du switch.
- Gestion d'un programme à l'aide d'un menu.
- Modélisation d'un problème simple sous forme informatique.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int x=0,choix;

do
    {
    cout<<"x vaut "<<x<<endl;
    cout<<"1 : Ajouter 1"<<endl;
    cout<<"2 : Multiplier par 2"<<endl;
    cout<<"3 : Soustraire 4"<<endl;
    cout<<"4 : Quitter"<<endl;
    cout<<"Votre choix : ";cin>>choix;

    switch(choix)
    {
    case 1 : x++;break;
    case 2: x=x*2; break;
    case 3: x=x-4;break;
    }
```

```
}while(choix!=4);

cout<<"La valeur finale de x vaut : "<<x<<endl;

return 0;
}
```

EXERCICE 11

Ecrire un programme qui demande à l'utilisateur de taper des entiers strictement positifs et qui affiche leur moyenne. Lorsqu'on tape une valeur négative, le programme affiche ERREUR et demande de retaper une valeur. Lorsqu'on tape 0, cela signifie que le dernier entier a été tapé. On affiche alors la moyenne. Si le nombre d'entiers tapés est égal à 0, on affiche PAS DE MOYENNE.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation d'un while de difficulté moyenne.
- Etude d'un algorithme usuel : calcul d'une moyenne.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int x, s=0,nb=0;
double moyenne;

do{
  cout<<"Tapez un entier :";cin>>x;
  if(x>0){s=s+x;nb++;}
    else if(x<0)cout<<"ERREUR ";

  }while(x!=0);

if(nb==0)cout<<"AUCUN ENTIER TAPE "<<endl<<"PAS DE MOYENNE"<<endl;
else {
     moyenne=(double)s/nb;
     cout<<"La moyenne vaut : "<<moyenne<<endl;
     }

return 0;
}
```

EXERCICE 12

Ecrire un programme qui demande à l'utilisateur de taper un entier N et qui calcule u(N) défini par :
u(0)=3
u(1)=2
u(n)=n.u(n-1)+(n+1).u(n-2)+n

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation d'un for.
- Etude d'un algorithme usuel : calcul d'une suite récurrente assez difficile.
- Modélisation d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>

int main()
{
int N,u,i=0,v,w;

cout<<"Tapez la valeur de N : ";cin>>N;
u=3;
v=2;
if(N==0)w=u;
else if(N==1)w=v;
else for(i=2;i<=N;i++){w=i*v+(i+1)*u+i;u=v;v=w;}

cout<<"u("<<N<<")="<<w<<endl;

return 0;
}
```

EXERCICE 13

Ecrire un programme qui demande de saisir 10 entiers et qui affiche le nombre d'occurrences de la note la plus haute.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation d'un for.
- Etude d'un algorithme usuel de difficulté moyenne : calcul du nombre d'occurence d'une valeur.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int nb,max,x,i;

for(i=0;i<10;i++)
    {
    cout<<"Tapez un entier : ";cin>>x;
     if(i==0){max=x;nb=1;}
    else if(x==max)nb++;
    else if(max<x){max=x;nb=1;}
    }

cout<<"le nombre d'occurences de "<<max<<" est "<<nb<<endl;

return 0;
}
```

EXERCICE 14

Ecrire un programme qui demande de saisir un entier N et qui affiche N!.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation du for.
- Etude d'un algorithme usuel : calcul d'une factorielle.
- Modélisation d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int N,i,f=1;

cout<<"Tapez un entier : ";cin>>N;
for(i=2;i<=N;i++)f=f*i;
cout<<N<<"! vaut "<<f<<endl;

return 0;
}
```

EXERCICE 15

Ecrire un programme qui demande de saisir un entier et qui indique si cet entier est premier ou non.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation d'un while de difficulté moyenne.
- Etude d'un algorithme usuel assez difficile : primarité d'un entier.
- Modélisation d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int n;
bool premier=true;
int d=2;

cout<<"Veuillez saisir un entier : ";cin>>n;

if(n<=1)premier=false;
    else
    {
    while(premier==true && d*d<=n)
        if(n%d==0)premier=false; else d=d+1;
    }
if(premier)cout<<n<<" est premier"<<endl;
    else cout<<n<<" n'est pas premier"<<endl;

return 0;
}
```

EXERCICE 16

Ecrire un programme qui demande à l'utilisateur de saisir un entier N et qui affiche le nombre de nombres premiers inférieurs ou égaux à N.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation des boucles for et while.
- Imbrication de boucles.
- Lire précisément un énoncé.
- Modélisation assez complexe d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;
```

```
int main()
{
int N,i,nb=0,d;
bool est_premier;

cout<<"Tapez la valeur de N : ";cin>>N;

for(i=2;i<=N;i++)
    {
    /* ecrire un programme qui teste si i est premier*/
    est_premier=true;
    d=2;
    while(est_premier && d*d<=i)
        if(i%d==0)est_premier=false; else d++;

    if(est_premier==true)nb++;
    }

cout<<"Le nombre de nombre premiers inférieurs ou égaux à "
    <<N<<" est "<<nb<<endl;

return 0;
}
```

EXERCICE 17

Ecrire un programme qui demande à l'utilisateur de saisir un entier N et qui affiche le N-ième nombre premier.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation des boucles for et while.
- Imbrication de boucles assez complexe.
- Lire précisément un énoncé.
- Modélisation assez complexe d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int N,i=1,nb=0,d;
bool est_premier;

cout<<"Tapez la valeur de N : ";cin>>N;

while(nb<N)
    {
    i++;
    est_premier=true;
    d=2;
    while(est_premier && d*d<=i)
        if(i%d==0)est_premier=false; else d++;

    if(est_premier==true)nb++;
```

```
    }

cout<<"Le N-ième nombre premier est "<<i<<endl;

return 0;
}
```

EXERCICE 18

Ecrire un programme qui demande à l'utilisateur de saisir un entier N et qui affiche la figure suivante.

```
N=1
*
N=2
**
*
N=3
***
**
*
```

et ainsi de suite

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation des boucles for.
- Imbrication de boucles assez complexe.

Voici le fichier source :

```
#include <iostream>

using namespace std;

int main()
{
    int N=0;
    cout<<"Saisissez une valeur pour N: ";
    cin>>N;
    cout<<endl<<endl;
    cout<<"N= "<<n<<endl;
    for(int i=0;i<N;i++)
    {

            for(int j=0;j<(N-i);j++)
            {
            cout<<"*";
            }
    cout<<endl;
    }
      cout << "Appuyez sur une touche pour continuer ..." << endl;
cin.ignore();
cin.get();
return 0;
}
```

EXERCICE 19

Ecrire un programme qui demande à l'utilisateur de saisir un entier N et qui affiche la figure suivante.

```
N=1
*
N=2
**
 *
N=3
***
 **
  *
```

et ainsi de suite

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation des boucles for.
- Imbrication de boucles.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int i,j,N;

cout<<"Tapez la valeur de N : ";cin>>N;

for(i=1;i<=N;i++)
    {
    for(j=1;j<i;j++)cout<<" ";
    for(j=1;j<=N+1-i;j++)cout<<"*";
    cout<<endl;
    }
return 0;
}
```

EXERCICE 20

On considère la suite hongroise : u(0)=a (a entier)
si u(n) pair alors u(n+1)=u(n)/2 sinon u(n+1)=3*u(n)+1

Pour toutes les valeurs a, il existe un entier N tel que u(N)=1 (conjecture admise).

a) Ecrire un programme qui demande à l'utilisateur de taper a et qui affiche toutes les valeurs de u(n) de n=1 à n=N.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation du while.
- Lire précisément un énoncé.
- Modélisation assez complexe d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int a,n,u;
cout<<"Tapez la valeur de a : ";cin>>a;
n=0;
u=a;

while(u!=1)
    {
    if(u%2==0)u=u/2; else u=3*u+1;
    n++;
    cout<<"u("<<n<<")="<<u<<endl;
    }
return 0;
}
```

b) Ecrire un programme qui demande à l'utilisateur de taper un entier M puis qui cherche la valeur de a comprise entre 2 et M qui maximise la valeur de N. On appelle A cette valeur. La programme doit afficher la valeur A et la valeur N correspondante.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation du while et du for.
- Imbrication de boucles.
- Lire précisément un énoncé.
- Modélisation assez complexe d'un problème issu des mathématiques.

Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int a,n,u,M,amax,nmax;
cout<<"Tapez la valeur de M : ";cin>>M;
amax=2;
nmax=2;

for(a=3;a<=M;a++)
    {
    n=0;
    u=a;
    while(u!=1)
       {
       if(u%2==0)u=u/2; else u=3*u+1;
```

```
        n++;
        }
    if(n>nmax){amax=a;nmax=n;}
    }
cout<<"La valeur de A est :"<<amax<<endl;
cout<<"La valeur de N correspondante est :"<<nmax<<endl;

return 0;
}
```

Les tableaux

Les tableaux

EXERCICE 1

Écrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit afficher le nombre d'entiers supérieurs ou égaux à 10.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple de tableaux.
- Un algorithme simple sur un tableau : recherche du nombre d'éléments vérifiant une propriété.

Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int t[10],i,nb=0;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>t[i];}
for(i=0;i<N;i++)if(t[i]>=10)nb++;
cout<<"Le nombre d'entiers supérieurs ou égaux à 10 est : "
    <<nb<<endl;
return 0;
}
```

EXERCICE 2

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V. Le programme doit rechercher si V se trouve dans le tableau et afficher "V se trouve dans le tableau" ou "V ne se trouve pas dans le tableau".

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Un algorithme simple sur un tableau : recherche d'un élément dans un tableau.

Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;
```

```
int main()
{
int t[N],i,V;
bool trouve;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>t[i];}
cout<<"Tapez la valeur de V : ";cin>>V;

trouve=false;
i=0;
while(!trouve && i<N)
    if(t[i]==V)trouve=true; else i++;
if(trouve) cout<<"La valeur V se trouve dans le tableau"<<endl;
    else cout<<"La valeur V ne se trouve pas dans le tableau"<<endl;
return 0;
}
```

EXERCICE 3

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau. Le programme doit ensuite afficher l'indice du plus grand élément.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple de tableaux.
- Un algorithme simple sur un tableau : recherche de l'indice du plus grand élément.

Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;
int main()
{
int t[N],i,indice;

for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>t[i];}
indice=0;
for(i=1;i<N;i++)
    if(t[indice]<t[i])indice=i;

cout<<"L'indice du plus grand élément est : "<<indice<<endl;
return 0;

}
```

EXERCICE 4

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V. Le programme doit rechercher si V se trouve dans le tableau et doit supprimer la première occurrence de V en décalant d'une case vers la gauche les éléments suivants et en rajoutant un 0 à la fin du tableau. Le programme doit ensuite afficher le tableau final.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple de tableaux.
- Un algorithme simple sur un tableau : supression d'un éléments avec décalage des suivants.

Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int t[N],i,j,V;
bool trouve;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>t[i];}
cout<<"Tapez la valeur de V : ";cin>>V;

trouve=false;
i=0;
while(!trouve && i<N)
    if(t[i]==V)trouve=true; else i++;

if(trouve)
    {
    for(j=i;j<N-1;j++)t[j]=t[j+1];
    t[N-1]=0;
    }
for(i=0;i<N;i++)cout<<t[i]<<endl;

return 0;
}
}
```

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de saisir 10 entiers stockés dans un tableau ainsi qu'un entier V et un entier i compris entre 0 et 9. Le programme doit décaler d'une case vers la droite tous les élements à partir de l'indice i (en supprimant le dernier élément du tableau) et doit mettre la valeur V dans le tableau à l'indice i. Le programme doit ensuite afficher le tableau final. {{Boîte déroulante|titre=Solution|contenu =

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Un algorithme simple sur un tableau : insertion dans en tableau avec décalage.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int t[N],i,indice,V;

for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>t[i];}
```

```
cout<<"Tapez un indice (de 0 à 9) : ";cin>>indice;
cout<<"Tapez la valeur de V : ";cin>>V;

if(indice>=0 && indice<=N-1)
    {
    for(i=N-1;i>indice;i--)t[i]=t[i-1];
    t[indice]=V;
    }

for(i=0;i<N;i++)cout<<t[i]<<endl;

return 0;
}
```

EXERCICE 6

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit ensuite afficher soit "le tableau est croissant", soit "le tableau est décroissant", soit "le tableau est constant", soit "le tableau est quelconque". {{Boîte déroulante|titre=Solution|contenu =

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Un algorithme simple sur un tableau : vérifier si le tableau vérifie une propriété donnée.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int a[N],i;
bool trouve=false;
bool croissant=true,decroissant=true;

for(i=0;i<N;i++)
    {
    cout<<"Veuillez taper l'entier numero "<<i<<" : ";cin>>a[i];
    }
for(i=0;i<N-1;i++)
    {
    if(a[i]>a[i+1])croissant=false;
    if(a[i]<a[i+1])decroissant=false;
    }

if(croissant && decroissant) cout<<"le tableau est constant"<<endl;
if(croissant && !decroissant) cout<<"le tableau est croissant"<<endl;
if(!croissant && decroissant) cout<<"le tableau est decroissant"<<endl;
if(!croissant && !decroissant) cout<<"le tableau est quelconque"<<endl;

return 0;
}
```

Solution 2:

```
// Par Ismail Zouaoui
#include <iostream>
```

```
using namespace std;

int const SIZE = 10;

int main()
{

    int table[SIZE];
    bool ordre = true;

    cout << "Entre 10 entiers: ";

    for(int i=0; i < SIZE; i++)
    {
        cin >> table[i];
    }

    if(table[0] < table[SIZE-1])
    {
        for(int i=0; i<SIZE-1; i++)
        {
            if(table[i] > table[i+1])
            {
                cout << "le tableau est quelconque.\n";
                ordre = false;
                break;
            }
        }
        if(ordre == true)
        {
            cout << "le tableau est croissant.\n";
        }
    }
    else if(table[0] > table[SIZE-1])
    {
        for(int i=0; i<SIZE-1; i++)
        {
            if(table[i] < table[i+1])
            {
                cout << "le tableau est quelconque.\n";
                ordre = false;
                break;
            }
        }
        if(ordre == true)
        {
            cout << "le tableau est decroissant.\n";
        }
    }
    else if(table[0] == table[SIZE-1])
    {
        for(int i=0; i<SIZE-1; i++)
        {
            if(table[i] != table[i+1])
            {
                cout << "le tableau est quelconque.\n";
                ordre = false;
                break;
            }
        }
        if(ordre == true)
        {
            cout << "le tableau est constant.\n";
        }
```

```
    }

    return 0;
}
```

EXERCICE 7

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.
Algorithme suggéré :
On cherche l'indice du plus petit élément parmi les indices de 0 à 9 et on échange cet élément avec t[0].
On cherche l'indice du plus petit élément parmi les indices de 1 à 9 et on échange cet élément avec t[1].
On cherche l'indice du plus petit élément parmi les indices de 2 à 9 et on échange cet élément avec t[2].
... On cherche l'indice du plus petit élément parmi les indices de 8 à 9 et on échange cet élément avec t[8].

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Un algorithme simple sur un tableau : tri d'un tableau.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int a[N],i,j,min,imin,tmp;

for(i=0;i<N;i++)
  {
  cout<<"Veuillez taper l'entier numero "<<i<<" : ";cin>>a[i];
  }

for(i=0;i<N-1;i++)
  {
  imin=i;min=a[i];
  for(j=i+1;j<N;j++)if(a[j]<min){min=a[j];imin=j;}

  tmp=a[imin];a[imin]=a[i];a[i]=tmp;
  }
cout<<"VOICI LE TABLEAU TRIE :"<<endl;
for(i=0;i<N;i++)cout<<"a["<<i<<"]="<<a[i]<<endl;

return 0;
}
```

EXERCICE 8

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers qui seront stockés dans un tableau. Le programme doit trier le tableau par ordre croissant et doit afficher le tableau.
Algorithme suggéré (tri bulle) :
On parcourt le tableau en comparant t[0] et t[1] et en échangeant ces éléments s'ils ne sont pas dans le bon ordre.

on recommence le processus en comparant t[1] et t[2],... et ainsi de suite jusqu'à t[8] et t[9].
On compte lors de ce parcours le nombre d'échanges effectués.
On fait autant de parcours que nécessaire jusqu'à ce que le nombre d'échanges soit nul : le tableau sera alors trié.

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Un algorithme simple sur un tableau : tri d'un tableau.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int a[N],i,nb,tmp;

for(i=0;i<N;i++)
{cout<<"Veuillez taper l'entier numero "<<i<<" : ";cin>>a[i];}

do
    {
    nb=0;
    for(i=0;i<N-1;i++)
        if(a[i]>a[i+1])
            {
            tmp=a[i];a[i]=a[i+1];a[i+1]=tmp;
            nb++;
            }
    }while(nb!=0);

cout<<"VOICI LE TABLEAU TRIE :"<<endl;
for(i=0;i<N;i++)cout<<"a["<<i<<"]="<<a[i]<<endl;

return 0;
}
```

EXERCICE 9

Ecrire un programme qui saisit 2 tableaux de 10 entiers a et b. c est un tableau de 20 entiers. Le programme doit mettre dans c la fusion des tableaux a et b. On copiera dans les 10 premières cases de c le tableau a, dans les dix dernières le tableau b. Le programme affiche ensuite le tableau c.

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Un algorithme simple sur un tableau : fusion de 2 tableaux.
- Voici le fichier source :

```
#include<iostream>
```

```
using namespace std;

const int N=10;

int main()
{
int a[N],b[N],c[2*N],i;

cout<<"SAISIE DU TABLEAU a"<<endl;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>a[i];}

cout<<"SAISIE DU TABLEAU b"<<endl;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>b[i];}

for(i=0;i<2*N;i++)if(i<N)c[i]=a[i];else c[i]=b[i-N];

cout<<"VOICI LE TABLEAU c"<<endl;
for(i=0;i<2*N;i++)cout<<c[i]<<" ";
cout<<endl;
return 0;
}
```

EXERCICE 10

Ecrire un programme qui saisit 2 tableaux de 10 entiers a et b qui doivent être triés dans l'ordre croissant. Le programme devra tout d'abord vérifier que les deux tableaux sont triés. Le tableau c est un tableau de 20 entiers. Le programme doit mettre dans c la fusion des tableaux a et b. Le tableau c devra contenir les éléments de a et ceux de b et devra être trié. Le programme affiche ensuite le tableau c.

Solution

```
#include<iostream>
using namespace std;

const int N=10;

int main()
{
int a[N],b[N],c[2*N],i,trie,indicea,indiceb;

cout<<"SAISIE DU TABLEAU a"<<endl;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>a[i];}

cout<<"SAISIE DU TABLEAU b"<<endl;
for(i=0;i<N;i++){cout<<"Tapez un entier ";cin>>b[i];}

trie=true;
i=0;
while(trie && i<N-1)if(a[i]>a[i+1])trie=false; else i++;

if(!trie)cout<<"Le tableau a n'est pas trié"<<endl;
    else
    {
    trie=true;
    i=0;
    while(trie && i<N-1)if(b[i]>b[i+1])trie=false; else i++;

    if(!trie)cout<<"Le tableau b n'est pas trié"<<endl;
        else
        {
        indicea=0;indiceb=0;
```

```
        for(i=0;i<2*N;i++)
            {
            if(indicea==N){c[i]=b[indiceb];indiceb++;}
            else if(indiceb==N){c[i]=a[indicea];indicea++;}
            else if(a[indicea]<b[indiceb]){c[i]=a[indicea];indicea++;}
            else {c[i]=b[indiceb];indiceb++;}
            }
        }
    }

cout<<"VOICI LE TABLEAU c"<<endl;
for(i=0;i<2*N;i++)cout<<c[i]<<" ";
cout<<endl;

return 0;
}
```

EXERCICE 11

Ecrire un programme qui gère une liste d'entiers grâce au menu suivant :
1. Ajouter un entier
2. Afficher la liste des entiers
3. Supprimer dernier entier de la liste.
4. Afficher la dernière note tapée
5. Quitter

Il y aura au maximum 10 entiers. Lorsqu'on rajoute un entier, il sera rajouté à la fin de la liste.

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Festion d'une liste simple grâce à un tableau statique.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;
int main()
{

int t[N],nb=0,choix,e,i;
bool fini=false;

while(fini==false)
    {
    cout<<"1. Ajouter un entier"<<endl;
    cout<<"2. Afficher la liste des entier"<<endl;
    cout<<"3. Supprimer le dernièr entier de la liste"<<endl;
    cout<<"4. Afficher le dernier entier de la liste"<<endl;
    cout<<"5. Quitter"<<endl;
    cout<<"Votre choix : ";cin>>choix;
    switch(choix)
        {
        case 1 : cout<<"Tapez un entier : ";cin>>e;
                if(nb<N){t[nb]=e; nb++; cout<<"ENTIER AJOUTE"<<endl;}
                else cout<<"IMPOSSIBLE LE TABLEAU EST PLEIN"<<endl;
        break;
```

```
        case 2 : if(nb==0)cout<<"LA LISTE EST VIDE"<<endl;
                    else {
                        cout<<"VOICI LA LISTE"<<endl;
                        for(i=0;i<nb;i++)cout<<t[i]<<" ";
                         cout<<endl;
                        }
        break;

        case 3 : if(nb>0){nb--; cout<<"ENTIER SUPPRIME"<<endl;}
                    else cout<<"LA LISTE EST VIDE"<<endl;
        break;

        case 4 : if(nb>0)cout<<"lLE DERNIER ENTIER EST "<<t[nb-1]<<endl;
                    else cout<<"LA LISTE EST VIDE"<<endl;
        break;

        case 5 : fini=true;
        break;
        }

    }
return 0;
}
```

EXERCICE 12

Ecrire un programme qui gère une liste d'entiers grâce au menu suivant :
1. Ajouter un entier
2. Afficher la liste des entiers
3. Supprimer le premier entier ayant une valeur donnée.
4. Supprimer tous les entiers ayant une valeur donnée
5. Quitter

Il y aura au maximum 10 entiers. La liste devra être en permanence triée : lorqu'on rajoute un entier, il sera inséré au bon endroit dans la liste pour que celle-ci reste triée.

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation simple de tableaux.
  - Gestion d'une liste triée grâce à un tableau statique.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=10;
int main()
{

int t[N],nb=0,choix,e,V,i,j,trouve;
bool fini=false;

while(fini==false)
    {
    cout<<"1. Ajouter un entier"<<endl;
    cout<<"2. Afficher la liste des entier"<<endl;
    cout<<"3. Supprimer le premier entier ayant une valeur donnée"<<endl;
    cout<<"4. Supprimer tous les entiers ayant une valeur donnée"<<endl;
    cout<<"5. Quitter"<<endl;
```

```
    cout<<"Votre choix : ";cin>>choix;
    switch(choix)
        {
        case 1 : if(nb<N)
                    {
                    cout<<"Tapez un entier : ";cin>>e;
                    i=0;
                    while(i!=nb && t[i]<e)i++;
                    for(j=nb;j>i;j--)t[j]=t[j-1];
                    t[i]=e;
                    nb++;
                    }
                 else cout<<"IMPOSSIBLE LE TABLEAU EST PLEIN"<<endl;
        break;

        case 2 : if(nb==0)cout<<"LA LISTE EST VIDE"<<endl;
                    else {
                        cout<<"VOICI LA LISTE"<<endl;
                        for(i=0;i<nb;i++)cout<<t[i]<<" ";
                         cout<<endl;
                        }
        break;

        case 3 : cout<<"Tapez la valeur à supprimer :";cin>>V;
                 trouve=false;
                 i=0;
                 while(!trouve && i<nb)if(t[i]==V)trouve=true; else i++;
                 if(trouve)
                    {
                    for(j=i;j<nb-1;j++)t[j]=t[j+1];
                    nb--;
                    }
        break;

        case 4 : cout<<"Tapez la valeur à supprimer :";cin>>V;
                 j=0;
                 for(i=0;i<nb;i++)
                     if(t[i]!=V){t[j]=t[i];j++;}
                 nb=j;
        break;

        case 5 : fini=true;
        break;
        }

    }
return 0;
}
```

EXERCICE 13

Ecrire un programme qui demande à l'utilisateur de taper un entier N<=20 et qui affiche la N-ième ligne du triangle de pascal.
ligne 1 : 1 1
ligne 2 : 1 2 1
ligne 3 : 1 3 3 1
ligne 4 : 1 4 6 4 1
et ainsi de suite ...

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :

- Utilisation simple de tableaux.
- Gestion d'une liste triée grâce à un tableau statique.

- Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
int a[21],i,j,N;
cout<<"Veuillez taper N : ";cin>>N;
for(i=1;i<=N;i++)
    {
    if(i==1)a[0]=1;
        a[i]=1;
    for(j=i-1;j>=1;j--)a[j]=a[j]+a[j-1];
    }
for(i=0;i<=N;i++)cout<<a[i]<<" ";
cout<<endl;
return 0;
}
```

EXERCICE 14

Ecrire un programme qui demande à l'utilisateur de taper 10 entiers compris entre 0 et 20 qui seront stockés dans un tableau et qui affiche le nombre de fois qu'on a tapé un 0, le nombre de 1, le nombre de 2, ..., le nombre de 20.

Solution

```
#include<iostream>
using namespace std;

int main()
{
int a[10],nb[21],i;

for(i=0;i<10;i++)
{
    do {cout<<"Veuillez taper l'entier numero "<<i<<" : ";cin>>a[i];}
        while (a[i]>20 || a[i]<0);
}

for(i=0;i<21;i++)nb[i]=0;
for(i=0;i<10;i++)nb[a[i]]++;

for(i=0;i<21;i++){cout<<"Il y a "<<nb[i]<<" fois l'entier "<<i<<endl;}
return 0;
}
```

EXERCICE 15

Ecrire un programme qui demande à l'utilisateur de taper le contenu d'un tableau de réels de 3 lignes et 3 colonnes et qui affiche ce tableau mais en affichant la moyenne des éléments de chaque ligne, de chaque colonne et la moyenne globale.

Solution

- Cet exercice a pour but de vérifier les points techniques suivants :
  - Utilisation de tableaux à 2 dimensions.
  - Modélisation d'un problème mathématique basique.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

const int N=3;
const int M=3;

int main()
{
double t[N][M],moyL[N],moyC[M],moy;
int i,j;

for(i=0;i<N;i++)
    for(j=0;j<M;j++)
    {
    cout<<"Tapez la valeur de la ligne "<<i<<" colonne "<<j<<" : ";
    cin>>t[i][j];
    }

for(i=0;i<N;i++)moyL[i]=0;
for(j=0;j<M;j++)moyC[j]=0;
moy=0;

for(i=0;i<N;i++)
    for(j=0;j<M;j++)
        {
        moyL[i]=moyL[i]+t[i][j];
        moyC[j]=moyC[j]+t[i][j];
        moy=moy+t[i][j];
        }

for(i=0;i<N;i++)moyL[i]=moyL[i]/N;
for(j=0;j<M;j++)moyC[j]=moyC[j]/M;
moy=moy/(N*M);

for(i=0;i<N;i++)
    {
    for(j=0;j<M;j++)
        cout<<t[i][j]<<" ";
    cout<<moyL[i]<<endl;
    }
for(j=0;j<M;j++)
    cout<<moyC[j]<<" ";
cout<<endl;

return 0;
}
```

Les tableaux de char

Les tableaux de char

EXERCICE 1

Écrire une fonction qui a en paramètres une chaîne de caractères (paramètre en entrée) et un entier e (paramètre en sortie). Cette fonction renvoie un booléen. La fonction renvoie true si la chaîne de caractères est un entier écrit sous la forme d'une suite de chiffres qui ne commence pas par 0, elle renvoie false sinon. Si la chaîne est correcte, la fonction renvoie dans e la valeur de cet entier.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- La validation d'une chaîne de caractères.
- Transformation d'une chaîne vers un autre type.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

bool valide_entier(char t[],int &e)
{
bool valide=true;
int i=0;
e=0;
if(t[0]<'1' || t[0]>'9')valide=false;

while(valide && t[i]!='\0')
{
if(t[i]>='0' && t[i]<='9'){e=10*e+(t[i]-'0');i++;}
else valide=false;
}
return valide;
}

int main()
{
int a;
char t[20];

do{cout<<"Tapez une chaine :";cin>>t;}while(!valide_entier(t,a));

cout<<"L'entier vaut :"<<a<<endl;
return 0;
}
```

EXERCICE 2

Ecrire une fonction qui a en paramètre une chaîne de caractères (paramètre en entrée et en sortie) et qui transforme toutes les minuscules de la chaîne en majuscules.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- Transformation d'une chaîne de caractrères.
- Majuscules et minuscules.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

void minmaj(char t[])
{
int i=0;
while(t[i]!='\0')
{
if(t[i]>='a' && t[i]<='z')t[i]=t[i]+('A'-'a');
i++;
}
}

int main()
{
char a[50];
cout<<"Tapez une chaine svp :";cin>>a;
minmaj(a);
cout<<"La chaine finale est :"<<a<<endl;
return 0;
}
```

EXERCICE 3

Ecrire une fonction qui a en paramètre une chaîne de caractères (paramètre en entrée et en sortie) et qui supprime toutes les voyelles.

Solution

```
- y est une voyelle dans ce programme
- la phrase peut contenir des espaces mais doit avoir moins de 80 caractères incluant le '\0'
- Si une voyelle à un accent, elle sera traité comme une consonne (tout comme une voyelle maju
- La voyelle est remplacée par un . pour afficher la suppression de la voyelle


#include <iostream>
using namespace std;
#include <conio.h>
//------------------------------------------------------------------------------

void remplaceVoyelle(char phrase[])
{
int i = 0;
while(phrase[i]!='\0')
 {
 if((phrase[i] == 'a')||(phrase[i] == 'e')||(phrase[i] == 'i')||(phrase[i] == 'o')||(phrase[i]
  {
  phrase[i]= '.';
  i++;
  }
 else
```

```
  {
  i++;
  }
 }
}


int main()
{
int taille = 80;
char phrase[taille];

cout <<"Entrez une phrase : ";
cin.get (phrase, taille);
//cout << "Voici la phrase entrée : "<<phrase<<endl;      //peut être activé

remplaceVoyelle(phrase);

cout<<"voici la phrase modifiee : "<<phrase <<endl;
cout<<"Appuyez sur une touche pour quitter le programme...";
getch();
return 0;
}
```

EXERCICE 4

Ecrire une fonction qui a en paramètres deux chaînes de caractères ch1 et ch2 (paramètres en entrée) et renvoie un booléen indiquant si la chaîne ch2 est contenue dans la chaîne ch1.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- Recherche d'une chaîne incluse dans une autre chaîne.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

bool contient(char ch1[],char ch2[])
{
int ii=0,i=0,j=0;
bool fini,trouve;
trouve=false;
fini=false;
while(!trouve && !fini)
{
if(ch1[ii]==ch2[j])
{
ii++;j++;
if(ch2[j]=='\0')trouve=true;
}
else
{
i++;ii=i;j=0;
}
if(ch1[ii]=='\0')fini=true;
}
return trouve;
}
```

```
int main()
{
char a[50],b[50];
cout<<"Tapez une chaine svp :";
cin>>a;
cout<<"Tapez une chaine svp :";
cin>>b;
if(contient(a,b))cout<<"la premiere chaine contient la seconde"<<endl;
else cout<<"la premiere chaine ne contient pas la seconde"<<endl;
return 0;
}
```

EXERCICE 5

Ecrire un programme qui demande à l'utilisateur de taper un verbe du premier groupe et qui le conjugue à l'indicatif présent.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- La construction d'une chaîne à partir d'une autre.
- Voici le fichier source :

```
#include<iostream>
using namespoace std;

int main()
{
char tt[20];
cout<<"Tapez un verbe du premier groupe : ";
cin>>tt;
int i;
i=strlen(tt);
if(i<=2 || tt[i-1]!='r' || tt[i-2]!='e')
cout<<"le verbe n'est pas du premier groupe"<<endl;
else
{
tt[i-2]='\0';
cout<<"je "<<tt<<"e"<<endl;
cout<<"tu "<<tt<<"es"<<endl;
cout<<"il "<<tt<<"e"<<endl;
cout<<"nous "<<tt<<"ons"<<endl;
cout<<"vous "<<tt<<"ez"<<endl;
cout<<"ils "<<tt<<"ent"<<endl;

}
return 0;
}
```

EXERCICE 6

Ecrire un programme qui saisit une chaîne pouvant contenir des espaces et qui affiche chaque mot de la chaîne, le séparateur étant l'espace.
Exemple, on tape : **je pense donc je suis**

Le programme affiche :
mot 1 : je
mot 2 : pense
mot 3 : donc
mot 4 : je
mot 5 : suis

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- Analyse syntaxique d'une chaîne de caractères.
- Voici le fichier source :

```
#include<iostream>
using namespace std;


int main()
{
char t[50];
char mot[50];

int i=0,j=0,state=1,num=0;

cout<<"Tapez une phrase : ";cin.getline(t,50);

while(t[i]!='\0'){
if(state==1)
{
if(t[i]!=' ')
{
     state=2;j=0;mot[0]=t[i];j++;
}
}
else
{
if(t[i]!=' '){mot[j]=t[i];j++;}
else
{
state=1;
num++;
mot[j]='\0';
cout<<"mot "<<num<<" : "<<mot<<endl;
}
}
i++;
}
if(state==2)
{
num++;
mot[j]='\0';
cout<<"mot "<<num<<" : "<<mot<<endl;
}
return 0;
}
```

EXERCICE 7

Ecrire un programme qui demande à l'utilisateur de taper une chaîne de caractères et qui indique si cette chaîne est un palyndrôme ou non.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- validation d'une chaîne.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
char t[50];
int i,j;
bool palyn;

palyn=true;
cout<<"Tapez une chaîne : ";cin>>t;
i=0;
j=strlen(t)-1;
while(palyn==true && i<j)
if(t[i]!=t[j])palyn=false; else {i++;j--;}

if(palyn) cout<<"C'est un palyndrome"<<endl;
else cout<<"ce n'est pas un palyndrôme"<<endl;
return 0;
}
```

EXERCICE 8

Ecrire un programme qui demande à l'utilisateur de taper une chaîne de caractères et qui affiche la lettre (minuscule ou majuscule) la plus fréquente.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- La manipulation de chaînes de caractères.
- Manipulation sur les majuscules ou minuscules.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

int main()
{
char ch[50];
int t[52];
int i,max;
char cmax;

cout<<"Tapez une chaîne : ";cin>>ch;
```

```
for(i=0;i<52;i++)t[i]=0;

i=0;
while(ch[i]!='\0')
{
if(ch[i]>='A' && ch[i]<='Z')t[ch[i]-'A']++;
else if(ch[i]>='a' && ch[i]<='z')t[ch[i]-'a'+26]++;
i++;
}

max=t[0];cmax='A';
for(i=1;i<52;i++)
if(max<t[i])
{
max=t[i];if(i<26)cmax=(char)(i+'A');
else cmax=(char)(i-26+'a');
}

cout<<"La lettre la plus fréquente est : "<<cmax<<endl;
return 0;
}
```

Les structures

Écrire un programme permettant de saisir les informations concernant N élèves sachant que :

- 10 < N < 30
- La structure contenant les informations de chaque élève doit comporter les champs suivants :
  - nom : chaine[10]
  - prenom : chaine[10]
  - note_ds : réel
  - coef_ds : entier
  - note_ex : réel
  - moyenne : réel
- on peut ranger ces structures d'informations dans un tableau.
- on doit afficher le nom,le prénom et la moyenne de chaque élève sachant que :

```
moyenne=((note_ds*coef_ds+note_ex*coef_ex)/(coef_ds+coef_ex)).
```

Les fonctions

Les fonctions

EXERCICE 1

Écrire une fonction distance ayant comme paramètres 4 doubles xa,ya et xb,yb qui représentent les coordonnées de deux points A et B et qui renvoie la distance AB. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Création de fonction simple.
- Passage de paramètres par valeur.
- Utilisation de return.
- Appel d'une fonction.
- Voici le fichier source :

```
#include<iostream>
using namespace std;
#include<cmath>

double distance(double xa, double ya, double xb, double yb)
{
double dx,dy;
dx=xa-xb;
dy=ya-yb;
return sqrt(dx*dx+dy*dy);
}

int main()
{
double x1,y1,x2,y2,d;

cout<<"Tapez l'abscisse de A : ";cin>>x1;
cout<<"Tapez l'ordonnée de A : ";cin>>y1;
cout<<"Tapez l'abscisse de B : ";cin>>x2;
cout<<"Tapez l'ordonnée de B : ";cin>>y2;

d=distance(x1,y1,x2,y2);

cout<<"La distance AB vaut : "<<d<<endl;
return 0;
}
```

EXERCICE 2

Ecrire une fonction f ayant comme paramètres un double x et un booléen ok et qui renvoie un double par un return. La fonction renvoie par un return la racine carrée de (x-1)*(2-x). La fonction renvoie par l'intermédiaire de la variable ok la valeur true si la fonction est définie au point x, false sinon. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Création de fonction simple.
- Passage de paramètres par valeur et par référence.
- Utilisation de return.
- Paramètres en entrées et en sorties d'une fonction.
- Appel d'une fonction.
- Jeu de tests d'une fonction.

Voici le fichier source :

```
#include<iostream>
using namespace std;
#include<cmath>

double f(double x, bool &ok)
{
double r=0;

if(x>=1 && x<=2){r=sqrt((x-1)*(2-x));ok=true;}
    else ok=false;
return r;
}

int main()
{
double x,y;
bool ok;
cout<<"Tapez x :";cin>>x;
y=f(x,ok);
if(ok)cout<<"f(x) vaut :"<<y<<endl;
    else cout<<"x n'est pas correct"<<endl;
return 0;
}
```

EXERCICE 3

Ecrire une fonction f ayant en paramètre un entier et qui renvoie par un return un booléen : true si l'entier est premier false sinon. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Création de fonction simple.
- Appel d'une fonction.
- Validation des données avant l'appel d'une fonction.
- Fonction renvoyant un booléen.
- Voici le fichier source :

```
#include<iostream>
using namespace std;
#include<cmath>

bool f(int x)
{
bool r=true;
```

```
int d=2;
while(r && d*d<=x)if(x%d==0)r=false; else d++;
return r;
}

int main()
{
int x;
bool premier;
do{
cout<<"Tapez x :";cin>>x;
}while(x<=0);
premier=f(x);
if(premier)cout<<"x est premier"<<endl;
    else cout<<"x n'est pas premier"<<endl;
return 0;
}
```

EXERCICE 4

Ecrire une fonction f ayant comme paramètre un entier n et qui renvoie le n-ième nombre premier : cette fonction utilisera la fonction du 3). Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Création de fonctions simples.
- Appel de fonction.
- Fonction qui appelle une autre fonction.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

bool premier(int x)
{
bool r=true;
int d=2;
while(r && d*d<=x)if(x%d==0)r=false; else d++;
return r;
}

int Npremier(int N)
{
int nb=0;
int i=2;

while(nb!=N)
    {
    if(premier(i))nb++;
    i++;
    }
return i-1;
}

int main()
{
int N,p;
cout<<"Tapez la valeur de N : ";cin>>N;
p=Npremier(N);
```

```
cout<<"Le N-ième nombre premier est : "<<p<<endl;
return 0;
}
```

EXERCICE 5

Ecrire une fonction swap ayant en paramètres 2 entiers a et b et qui echange les contenus de a et de b. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Création de fonction simple.
- Appel d'une fonction.
- Passage de paramètres par références.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

void swap(int &x, int &y)
{
int temp;
temp=x;
x=y;
y=temp;
}

int main()
{
int a,b;
cout<<"Tapez a :";cin>>a;
cout<<"Tapez b :";cin>>b;
swap(a,b);
cout<<"a vaut : "<<a<<endl;
cout<<"b vaut : "<<b<<endl;

return 0;
}
```

EXERCICE 6

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f doit renvoyer par un return un booleén b indiquant s'il existe une valeur comprise entre 0 et 10 dans les n premières cases du tableau t. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Ecriture d'une fonction ayant comme paramètre un tableau de taille quelconque.
- Recherche d'un élément dans un tableau vérifiant une propriété.
- Utilisation de return.

- Voici le fichier source :

```
#include<iostream>
using namespace std;

void saisir(int t[],int n)
{
int i;
for(i=0;i<n;i++)
{
cout<<"Tapez la valeur numero "<<i<<" : ";
cin>> t[i];
}
}

bool f(int t[], int n)
{
bool trouve=false;
int i=0;
while(!trouve && i<n)
if(t[i]>=0 && t[i]<=10)trouve=true; else i++;
return trouve;
}

int main()
{
int a[10];
saisir(a,10);
bool b;

b=f(a,10);
if(b)cout<<"Il existe une valeur entre 0 et 10"<<endl;
else cout<<"Il n'existe pas de valeurs entre 0 et 10"<<endl;
return 0;
}
```

EXERCICE 7

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f doit renvoyer par un return le nombre de valeurs comprises entre 0 et 10 dans les n premières cases du tableau t.Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Ecriture d'une fonction ayant comme paramètre un tableau de taille quelconque.
- Compter le nombre d'éléments dans un tableau vérifiant une propriété.
- Utilisation de return.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

void saisir(int t[],int n)
{
int i;
for(i=0;i<n;i++)
{
```

```
cout<<"Tapez la valeur numero "<<i<<" : ";
cin>> t[i];
}
}

int f(int t[], int n)
{
int nb=0;
int i;
for(i=0;i<n;i++)
if(t[i]>=0 && t[i]<=10)nb++;
return nb;
}

int main()
{
int a[10];
saisir(a,10);
int x;
x=f(a,10);
cout<<"Il y a "<<x<<" valeur(s) entre 0 et 10"<<endl;
return 0;
}
```

EXERCICE 8

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f possède un autre paramètre v , entier passé par référence. f doit renvoyer par un return un booleén b indiquant s'il existe une valeur comprise entre 1 et 10 dans les n premières cases du tableau t. Si f renvoie true, v est égal à la valeur de la première case du tableau comprise entre 0 et 10. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Ecriture d'une fonction ayant comme paramètre un tableau de taille quelconque.
- Recherche d'un élément dans un tableau.
- Utilisation de return.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

void saisir(int t[],int n)
{
int i; for(i=0;i<n;i++)
{
cout<<"Tapez la valeur numero "<<i<<" : ";
cin>> t[i];
}
}

bool f(int t[], int n, int &v)
{
bool trouve=false;
int i=0;
while(!trouve && i<n)
if(t[i]>=0 && t[i]<=10){trouve=true; v=t[i];}else i++;
return trouve;
```

```
}

int main()
{
int a[10];
bool b;
int w;

saisir(a,10);
b=f(a,10,w);
if(b)cout<<"Il existe une valeur entre 0 et 10 : "<<w<<" est la première de ces valeurs."<<end
else cout<<"Il n'existe pas de valeurs entre 0 et 10"<<endl;
return 0;
}
```

EXERCICE 9

Ecrire une fonction f ayant en paramètres un tableau t1 de taille quelconque et un entier n indiquant la taille du tableau, ainsi qu'un tableau t2 de la même taille que t1. f doit renvoyer par un return un entier nb indiquant le nombre de valeurs comprises entre 0 et 10 dans le tableau t1. f doit mettre dans le tableau t2 les différentes valeurs comprise entre 0 et 10 qu'il a rencontrées dans le tableau t1.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Ecriture d'une fonction ayant comme paramètre un tableau de taille quelconque.
- Extraction d'un sous-liste d'éléments.
- Utilisation de return.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

void saisir(int t[],int n)
{
int i;
for(i=0;i<n;i++)
    {
    cout<<"Tapez la valeur numero "<<i<<" : ";
    cin>> t[i];
    }
}

void afficher(int t[],int n)
{
int i;
for(i=0;i<n;i++)
    cout<<t[i]<<" ";
cout<<endl;
}

int f(int t1[], int n,int t2[])
{
int i=0,nb=0;

for(i=0;i<n;i++)if(t1[i]>=0 && t1[i]<=10){t2[nb]=t1[i];nb++;}
return nb;
}
```

```
int main()
{
int a[10],b[10];
int nb;

saisir(a,10);
nb=f(a,10,b);
cout<<"VOICI LES VALEURS ENTRE 0 ET 10 : "<<endl;
afficher(b,nb);
return 0;
}
```

EXERCICE 10

Ecrire une fonction f ayant en paramètres un tableau t de taille quelconque et un entier n indiquant la taille du tableau. f doit renvoyer par un return un entier égal à l'indice de la première case du tableau (parmi les n premières) comprise entre 0 et 10. S'il n'existe pas de telle valeur, la fonction renvoie -1. Tester cette fonction.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Ecriture d'une fonction ayant comme paramètre un tableau de taille quelconque.
- Recherche d'un élément dans un tableau.
- Utilisation de return.
- Voici le fichier source :

```
#include<iostream>
using namespace std;

void saisir(int t[],int n)
{
int i; for(i=0;i<n;i++)
{
cout<<"Tapez la valeur numero "<<i<<" : ";
cin>> t[i];
}
}

int f(int t[], int n)
{
int i=0,ind=-1;

while(ind==-1 && i<n)
if(t[i]>=0 && t[i]<=10)ind=i;else i++;
return ind;
}

int main()
{
int a[10];
int w;

saisir(a,10);
w=f(a,10);
if(w!=-1)cout<<"Il existe une valeur entre 0 et 10. "
<<"l'indice de la première case est "<<w<<endl;
else cout<<"Il n'existe pas de valeurs entre 0 et 10"<<endl;
return 0;
```

```
}
```

Les classes

EXERCICE 1 : rectangle

Écrire un programme utilisant une classe rectangle dont le constructeur prend deux paramètres, largeur et hauteur et qui offre les fonctions suivantes :

- calcul du périmètre
- calcul de la surface
- affichage

ainsi que les accesseurs et mutateurs triviaux (lecture et modification de la largeur et de la hauteur).

Solution

Découverte et travail des classes. Utilisation d'un constructeur et d'un destructeur.

```
#include<iostream>
#include <cstdlib>

using namespace std;

class Rectangle
{
public:
  Rectangle(unsigned int initLargeur, unsigned int initHauteur);
  ~Rectangle();
  unsigned int getLargeur() const { return largeur; };
  unsigned int getHauteur() const { return hauteur; };
  unsigned int perimetre() const { return 2*(largeur+hauteur); };
  unsigned int surface() const { return largeur * hauteur; };
  void setLargeur(unsigned int newLargeur) { largeur = newLargeur; };
  void setHauteur(unsigned int newHauteur) { hauteur = newHauteur; };
  void afficher();

private:
  unsigned int largeur;
  unsigned int hauteur;
};

Rectangle::Rectangle(unsigned int initLargeur, unsigned int initHauteur)
{
  largeur = initLargeur;
  hauteur = initHauteur;
}

Rectangle::~Rectangle()
{
}

void Rectangle::afficher()
{
  for(unsigned int i=0; i < hauteur; i++)
  {
    for(unsigned int j=0; j < largeur; j++)
      cout << "*";
    cout << endl;
  }
}
```

```
int main()
{
  Rectangle monRectangle(0,0);
  char choix = '0';
  unsigned int value;

  while(true)
  {
    do
    {
      cout << " Rectangle - Menu" << endl;
      cout << "1 - Modifier largeur du rectangle" << endl;
      cout << "2 - Modifier hauteur du rectangle" << endl;
      cout << "3 - Calculer les propriétés du rectangle" << endl;
      cout << "4 - Afficher le rectangle" << endl;
      cout << "5 - Quitter" << endl;

      cin >> choix;
    }while(choix < '1' || choix > '5');

    switch(choix)
    {
      case '1':
        cout << "Nouvelle largeur : ";
        cin >> value;
        monRectangle.setLargeur(value);
        break;
      case '2':
        cout << "Nouvelle hauteur : ";
        cin >> value;
        monRectangle.setHauteur(value);
        break;
      case '3':
        cout << "Périmètre : " << monRectangle.perimetre() << endl;
        cout << "Surface : " << monRectangle.surface() << endl;
        break;
      case '4':
        monRectangle.afficher();
        break;
      case '5':
        exit(0);
        break;
      default:
        cout << "Erreur ! Choix invalide." << endl;
        exit(1);
    }
  }

  return 2;
}
```

EXERCICE 2 : pile d'entiers

Une pile est un ensemble dynamique d'éléments où le retrait se fait d'une façon particulière. En effet, lorsque l'on désire enlever un élément de l'ensemble, ce sera toujours le dernier inséré qui sera retiré. Un objet pile doit ré pondre aux fonctions suivantes:

-Initialiser une pile

-Empiler un élément sur la pile (push)

-Dépiler un élément de la pile (pop)

pour cela nous allons supposer que les éléments à empiler sont de type int.

Le programme main comprend la définition d'une classe pile et un programme de test qui crée deux piles p1 et p2,empile dessus des valeurs

entières et les dépiler pour vérifier les opérations push et pop.

Solution

Découverte et travail des classes. Utilisation d'un constructeur et d'un destructeur.

```
#include<iostream.h>
class pile
{
int *tab;
int k; //indice de la 1er position vide
int max;
pile (int taille =100);
~pile();
int push  int x;  //empiler X
int  full();
int empty ();
int pop; int &x; //dépiler le dernier éllement
.
.
.
.
}
```

EXERCICE 3 : Fichier

Imaginons une application qui traite des fichiers. Ces fichiers vont être lus en mémoire, traités puis sauvegardés. Une fois lu en mémoire, un fichier a deux caractéristiques, une adresse à partir de laquelle se situe le fichier et une longueur, ce qui se concrétisera par un pointeur et une longueur en nombre d'octets. Imaginons la classe "Fichier" avec un constructeur et un destructeur et les trois méthodes suivantes:

-la méthode "Creation" qui va allouer un certain espace à partir du pointeur P,

-la méthode "Remplit" qui va remplir arbitrairement cet espace (ces remplissages arbitraires sont la preuve

de la bonne gestion mémoire car l'accès à une zone non déclarée provoque une violation d'accès),

-la méthode "Affiche" qui va afficher la zone mémoire pointée par P.

Puis écrivons un programme maître qui instancie notre classe par new, appelle nos trois méthodes et détruit l'objet par delete.

Solution

Découverte et travail des classes. Utilisation d'un constructeur et d'un destructeur.

```
#include <iostream.h>
// déclaration de la classe Fichier
class Fichier
{
char* P;
unsigned int Lg;
public:
Fichier();
~Fichier();
bool Creation(unsigned int);
void Remplit();
void Affiche();
};
// constructeur
Fichier::Fichier()
{
P=NULL;
Lg=0;
}
// destructeur
Fichier::~Fichier()
{
delete P;
}
// méthode Creation
bool Fichier::Creation(unsigned int L)
```

```
{
if((P=(char*)malloc(L))==NULL) return false;
Lg=L;
return true;
}
// Méthode Remplit
void Fichier::Remplit()
{
for(unsigned int i=0;i<Lg;i++) P[i]='a';
}
// Méthode Affiche
void Fichier::Affiche()
{
for(unsigned int i=0;i<Lg;i++) cout<<P[i];
}
//-----Programma maître (main)--------------
void main(void)
{
Fichier* f=new Fichier();
if (f->Creation(10))
{
f->Remplit();
f->Affiche();
}
delete f;
}
```

EXERCICE 4 : classe point

Réaliser **une classe point** permettant de manipuler un point d'un plan.on prévoira :

-un point est définit par ses coordonnées x et y (des membres privés)

-un constructeur (vous pouvez également implémenter les trois types de constructeur)

-une fonction membre déplace effectuant une translation définie par ses deux arguments dx et dy (double)

-une fonction membre affiche se contentant d'afficher les coordonnées cartésiennes du point.

-une fonction membre saisir se contentant de saisir les coordonnées cartésiennes du point.

-une fonction membre distance effectuant calculant la distance entre deux point.

-une fonction membre milieu donnant le milieu d'un segment.

on écrira séparément:

-un ficher source constituant la déclaration et la définition de la classe.

-un petit programme d'essai (main) gérant la classe point.

Solution

Découverte et travail des classes. Utilisation d'un constructeur et d'un destructeur.

```
//ficher source
#include <iostream.h>
#include <math.h>
class point
{
    double x;
    double y;
public:
    point();
    point(double,double);
    point(point &);
    ~point();
double get_x();
double get_y();
void set_x(double x1);
void set_y(double y1);
point deplace(double dx, double dy);
void affiche();
void saisir();
double distance(point &);
point milieu(point &);

};

point::point()
{}
point::point(double a,double b)
{
x=a;
y=b;
}
point::point(point &p)
{
    set_x(p.get_x());
    set_y(p.get_y());
}
point::~point()
{}
double point::get_x()
{
    return x;
}
double  point::get_y()
```

```
{
    return y;
}
void point::set_x(double a)
{
    x=a;
}
void point::set_y(double a)
{
    y=a;
}
point point::deplace(double dx,double dy)
{

    set_x(get_x()+dx);

    set_y(get_y()+dy);
    return *this;

}
double point::distance (point &p)
{
    double p1,x1,x2;
    x1=(get_x()-p.get_x())*(get_x()-p.get_x());
    x2=(get_y()-p.get_y())*(get_y()-p.get_y());
    //p1=sqrt(((get_x()-p.x)*((get_x()-p.x))+((get_y()-p.y)*(get_y()-p.y)));
    p1=sqrt(x1+x2);
    return p1;
}
void point::affiche()
{
    cout<<"les coordonnées sont:"<<endl;
    cout<<"x="<<get_x()<<endl;
    cout<<"y="<<get_y()<<endl;
}
void point::saisir()
{
    cout<<"donnée les coordonnées:"<<endl;
    cout<<"x="<<endl;
    cin>>x;
    cout<<"y="<<endl;
    cin>>y;
}
point point::milieu(point &p)
{
    point p1;
    p1.x=(get_x()+p.get_x())/2;
    p1.y=(get_y()+p.get_y())/2;
    return p1;
}

//programme d'essai(main)

#include<iostream.h>
#include "pointt.h"
void main()
{point p(1,1);
point x(5,5);
point c;

    p.affiche();
    p.deplace(5,5);
    p.affiche();
    cout<<"la distance est : "<<p.distance(x);
    c=p.milieu(x);
    c.affiche();
```

```
}
```

Exercice 5 : classe *Compteur*

On souhaite implémenter une classe représentant un compteur entier. Un tel objet se caractérise par :

Une valeur entière, positive ou nulle, nulle à l'origine.

Le fait qu'il ne peut varier que par pas de 1 (incrémentation ou décrémentation). On convient qu'une décrémentation d'un compteur nul est

sans effet.

Il s'agit de créer une classe Compteur pour rendre le service demandé. On écrira en outre un petit programme de test qui :

1. créera un compteur et affichera sa valeur;

2. l'incrémentera 10 fois, puis affichera à nouveau sa valeur;

3. le décrémentera 20 fois, puis affichera une troisième fois sa valeur

La sortie de ce programme doit donner (quelque chose comme) "0 10 0"

Solution

Découverte et travail des classes. Utilisation d'un constructeur et d'un destructeur.

```
Fichier Compteur.h : déclaration de la classe Compteur
class Compteur {  protected:  /* i.e. visible des classes dérivées */
  int _valeur;  public:              /* i.e. l'interface de la classe */
  Compteur();                    /* le constructeur, invoque a la création */
  void incr();
  void decr();
  int valeur();                  /* permet un accès "read-only" a l'attribut */
};
Fichier Compteur.C : implémentation de la classe Compteur

#include <iostream.h>
#include "Compteur.h"

/*************************************************************/
Compteur :: Compteur() {
  _valeur = 0;
}

/*************************************************************/
void Compteur :: incr() {
  _valeur ++;
}

/*************************************************************/
```

```
void Compteur :: decr() {
  if (_valeur>0) _valeur--;
}

/*************************************************************/
int Compteur :: valeur() {
  return _valeur;
}

Fichier testCompteur.C : programme principal

#include <iostream.h>
#include "Compteur.h"

/*************************************************************/
int main () {

  Compteur c;                    // automatiquement égal a 0 par le constructeur

  cout << c.valeur() << endl;
  for (int i=0; i<10; i++)
    c.incr();
  cout << c.valeur() << endl;    // le compteur est passe à 10
  for (int i=0; i<20; i++)       // "i" est local au "for" !!!
    c.decr();
  cout << c.valeur() << endl;    // le compteur est redescendu à 0

  return 0;                      // pour l'OS
}
```

Exercice 6

1/ On voudrait gérer les étudiants d'une institution à l'aide d'une classe Etudiant définie par :

les attributs suivants :

- nom : nom d'un étudiant

- prénom: prénom d'un étudiant

- tabnotes : tableau contenant les notes d'un étudiant, sachant qu'un étudiant a au total 10 notes.

les méthodes suivantes :

- void saisie (), permettant la saisie d'un étudiant

- void affichage (), permettant l'affichage d'un étudiant

- float moyenne (), retourne comme résultat la moyenne des notes d'un étudiant.

- int admis (), retourne comme résultat la valeur 1, si un étudiant est admis et la valeur 0, sinon. Un étudiant est considéré comme étant

admis lorsque la moyenne de ses notes est supérieure ou égale à 10.

- int exae_quo (Etudiant E), retourne comme résultat la valeur 1, si deux étudiants ont la même moyenne et la valeur 0, sinon.

Ecrire la classe Etudiant dans le langage C++.

2/ On voudrait maintenant représenter, à l'aide d'une nouvelle classe Etudiant_en_Maitrise, certains étudiants particuliers dans cette

institution qui sont les étudiants en dernière année d'études. Ces étudiants possèdent en effet un attribut supplémentaire : note_memoire,

qui représente la note de leur mémoire de fin d'études.

Les méthodes à associer à cette classe sont les suivantes :

- void saisie (), permettant la saisie d'un étudiant en maîtrise

- void affichage (), permettant l'affichage d'un étudiant en maîtrise

- float moyenne (), retourne comme résultat la moyenne des notes d'un étudiant en maîtrise

- int admis (), retourne comme résultat la valeur 1, si un étudiant est admis et la valeur 0, sinon. Un étudiant en maîtrise est considéré

comme étant admis lorsque, d'une part, la moyenne de ses notes est supérieure ou égale à 10 et d'autre part la note obtenue pour son mémoire

de fin d'études est supérieure ou égale à 10.

- int exae_quo (Etudiant_en_Maitrise E), retourne comme résultat la valeur 1, si deux étudiants ont d'une part la même moyenne et d'autre

part, la même note de mémoire et retourne la valeur 0, sinon.

a) Quelles sont les méthodes qui sont à redéfinir dans la classe Etudiant_en_Maitrise ?

b) Ecrire la classe Etudiant_en_Maitrise dans le langage C++.

Solution

Découverte et travail des classes. Utilisation d'un constructeur et d'un destructeur.

```
1/
class Etudiant
{ private:
            char nom[50], prenom[50];
            float tabnotes[10] ;
  public :
            void saisie () ;
```

```
            void affichage () ;
            float moyenne() ;
            int admis() ;
            int exae_quo (Etudiant E) ;
} ;
void Etudiant ::saisie ()
{ int i ;
            cout << "Donner le nom :" ;
            cin >> nom ;
            cout << "Donner le prénom :" ;
            cin >> prenom ;
            cout << "Saisie des notes \n" ;
            for (i = 0 ; i < 10 ; i++)
            {
                        cout << "Donner la note N°" << i<< " : " ;
                        cin >> tabnotes[i] ;
            }
}
void Etudiant ::affichage ()
{ int i ;
            cout << "Le nom :"<<nom<< endl ;
            cout << "Le prénom :" <<prenom<< endl ;
            for (i = 0 ; i < 10 ; i++)
                        cout << "La note N°" << i << "est " << tabnotes[i]<< endl ;
}
float Etudiant ::moyenne()
{ int i ;
  float som = 0;
            for (i = 0 ; i < 10 ; i++)
                        som + = tabnotes[i] ;
            return (som/10)
}
int Etudiant ::admis()
{ if (moyenne() >= 10) return (1); else return (0);}
```

```
int Etudiant ::Exae_quo(Etudiant E)
{ if (moyenne() == E.moyenne()) return (1); else return (0);}
2/
a) Les méthodes qui sont à redéfinir dans la classe Etudiant_en_Maitrise sont : saisie, affich
b)
class Etudiant_en_Maitrise : public Etudiant
{ private:
            float note_memoire ;
  public :
            void saisie () ;
            void affichage () ;
            int admis() ;
            int exae_quo (Etudiant_en_Maitrise E) ;
} ;
void Etudiant_en_Maitrise ::saisie ()
{          Etudiant ::saisie () ;
cout << "Donner la note du mémoire :" ;
            cin >> note_memoire ;
}
void Etudiant_en_Maitrise ::affichage ()
{          Etudiant :: affichage () ;
cout << "La note du mémoire :" << note_memoire<< endl ;
}
int Etudiant_en_Maitrise ::admis()
{ if ((moyenne() >= 10) && (note_memoire >=10))return (1); else return (0);}
int Etudiant_en_Maitrise ::Exae_quo(Etudiant E)
{ if ((moyenne() == E.moyenne()) && (note_memoire == E.note_memoire)) return (1); else return
```

Les templates

Exercice 1

[MOYEN]

Requis:

- manipulation de tableaux,
- savoir créer une fonction template.

Créez une fonction "trier" pour trier un tableau de 10 données avec un tri par sélection (https://fr.wikipedia.org/wiki/Tri_par_s%C3%A9lection) [archive]. Le tableau devra être de n'importe quel type. Cette fonction utilisera une autre fonction, "échanger", pour échanger les éléments du tableau.

Aide

Le tri par sélection scan un tableau et met sa valeur minimale au début. Il répète l'opération jusqu'à obtenir un tableau trié.

Solution

Voici le code source:

```
template<class T> T * échanger(T tab[10], int a, int b)
{
T c = tab[a];
tab[a] = tab[b];
tab[b] = c;
return tab;
}

template<class T>T * trier(T tab[10])
{
int j, min;
for(int i = 0; i < 9; i++)
{
for(j = i; j < 10; j++)
{
if(j == i)
min = j;
if(tab[j] < tab[min])
min = j;
}
échanger(tab, i, min);
}
return tab;
}
```

Exercice 2 : Fonction Template

[FACILE] Requis : Savoir créer et utiliser une fonction Template

1) Créer une fonction Template qui permettra de faire une somme entre un int et un float,

2) faire une spécialisation de la fonction en std::string, qui retournera la somme de la taille des deux chaines.

Exercice 3 : Classe Template

[FACILE-MOYEN] Requis : Savoir créer une classe Template

1) Créer une classe Template "Rectangle" prenant en paramètre les coordonnées de celui-ci sur un graphique(x,y). Evidemment, le type des attributs et des méthodes renvoyant une valeur (si il y en a) dépendront donc du paramètre Template rentré au préalable.

2) Créer une méthode retournant sa hauteur et son aire.

---AIDE :---

Voici comment déclarer votre Objet Rectangle dans le main() : Rectangle<float> myRectangle(1.2, 2.6, 1.5, 4.1);

Exercice 4

[MOYEN-DIFFICILE]

Requis:

- savoir utiliser les pointeurs et l'allocation dynamique de mémoire,
- savoir créer une classe template,
- savoir créer une structure.

Créez une classe liste simplement chaînée, avec une classe liste. Cette classe a un pointeur sur le premier élément de la liste. Elle a une méthode pour ajouter ou supprimer un élément au début de la liste et une pour afficher la liste en entier. Evitez toute fuite mémoire. Les éléments de la liste seront contenu dans la structure element.

Aide

Utilisez l'instruction `new` et `delete` pour supprimer et créer de nouveaux éléments. Une liste simplement chaînée est une structure de donnée. Elle est constituée de blocs où chaque bloc pointe sur le bloc suivant. Chaque bloc contient une donnée. Il est conseillé de réviser les pointeurs.

Solution

Voici le fichier source décomposé pour plus de lisibilité:

En-tête

```
#include <iostream>
using namespace std;
```

Structure élément

```
// Structure élément
template<class T> element
{
T var;
element<T> * suivant;
};
```

Classe liste

```
// Classe liste
Template<class T>class liste
{
public:
element<T> * début;
```

Constructeur et destructeur

```
liste()
{
début = NULL;
}

~liste()
{
while(début != NULL)
{
supprimer();
}
}
```

Ajouter et supprimer

```
void ajouter(T var)
{
if(début == NULL)
{
début = new element<T>;
début->suivant = NULL;
}
else
{
element<T> * p = new element<T>;
p->suivant = début;
```

```
début = p;
}
début->var = var;
}

void supprimer()
{
if(début == NULL)
return;
element<T> * p = début;
début = début->suivant;
delete p;
}
```

Afficher

```
void afficher()
{
element<T> * p = début;
while(p != NULL)
{
cout << p->var << endl;
p = p->suivant;
}
}
```

```
};
```

La STL

La STL

EXERCICE 1 : La classe string

Soit une chaîne de caractères contenant une date (JJ/MM/AAAA) et une heure (HH:NN) sous la forme `JJMMAAAAHHNN`. Par exemple `010920091123` représente la date du 1er septembre 2009 à 11h23.

Créer un programme permettant d'extraire les différents champs et de les afficher.

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- Manipulation des chaînes de caractères.

Voici le fichier source :

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

void afficherDateEtHeure(const string& s)
{
    if ( s.length() != 12 )
        cerr << "Chaine invalide." << endl;
    else
    {
        cout << "Date  : " << s.substr(0,2) << "/" << s.substr(2,2) << "/" << s.substr(4,4) <<
        cout << "Heure : " << s.substr(8,2) << "h" << s.substr(10,2) << endl;
    }
}

int main(int argc, char** argv)
{
    string s("010920091123");
    afficherDateEtHeure(s); // exemple
}
```

Les fichiers

Les fichiers : Lecture et écriture en mode texte

EXERCICE 1

Écrire un programme qui écrit dans le fichier example.txt le texte:

```
Hello world!
Voici un programme illustrant l'écriture dans un fichier
```

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- L'ouverture d'un fichier en écriture
- Tester si un fichier est ouvert (en particulier si vous avez les droits d'écriture sur le fichier)
- Fermer le fichier une fois l'écriture terminée

Voici le fichier source :

```
#include <iostream>
#include <fstream>

int main (int argc, char * argv[]) {
  std::ofstream myfile;
  char * filename = "example.txt";
  myfile.open (filename, std::ios::out);
  if(myfile.is_open())
  {
      myfile << "Hello world \n";
      myfile << "Voici un programme illustrant l'écriture dans un fichier \n";
  }
  else
  {
      std::cout << "Erreur à l'ouverture du fichier "<< filename << std::endl;
  }
  myfile.close();
  return 0;
}
```

EXERCICE 2

Ecrire un programme qui lit le fichier example.txt définit dans l'exemple précédent et affiche son contenu :
Vous devriez obtenir :

```
Hello world!
Voici un programme illustrant l'écriture dans un fichier
```

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- L'ouverture d'un fichier en lecture
- Tester si un fichier est ouvert (en particulier si vous avez les droits de lecture sur le fichier)
- Parcourir dans son intégralité un fichier
- Fermer le fichier une fois la lecture terminée

Voici le fichier source :

```
#include <iostream>
#include <fstream>
#include <string>

int main () {
  std::string line;
  char * filename = "example.txt";
  std::ifstream myfile (filename, std::ios::in);
  if (myfile.is_open())
  {
    while (! myfile.eof() )
    {
      getline (myfile,line);
      std::cout << line << std::endl;
    }
    myfile.close();
  }

  else std::cout << "Erreur à l'ouverture du fichier " << filename << std::endl;;

  return 0;
}
```

Les fichiers : Lecture et écriture en mode binaire

EXERCICE 1

Proposez un programme qui écrit en binaire une chaîne de caractère suivant de la liste des entiers de 0 à 1000 :

```
Liste des entiers de 1 à 1000
0
1
2
...
1000
```

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- L'ouverture d'un fichier en écriture en mode binaire
- Tester si un fichier est ouvert (en particulier si vous avez les droits d'écriture sur le fichier)
- Ecrire des données en binaire
- Fermer le fichier une fois l'écriture terminée

Voici le fichier source :

```
#include <iostream>
#include <fstream>

int main (int argc, char * argv[]) {
  std::ofstream myfile;
  char * filename = "exampleBinary.txt";
  myfile.open (filename, std::ios::out | std::ios::binary);
  if(myfile.is_open())
  {
      myfile.write("Liste des entiers de 1 à 1000",29*sizeof(char));
      for(int i = 0 ; i <= 1000 ; i++)
      {
          myfile.write((char*)&i,sizeof(int));
      }
  }
  else
  {
      std::cout << "Erreur à l'ouverture du fichier "<< filename << std::endl;
  }
  myfile.close();
  return 0;
}
```

EXERCICE 2

Proposez un programme qui lit le fichier écrit en binaire de l'exercice précédent et affiche son contenu. Vous devriez obtenir :

```
Liste des entiers de 1 à 1000
0
1
2
...
1000
```

Solution

Cet exercice a pour but de vérifier les points techniques suivants :

- L'ouverture d'un fichier en lecture en mode binaire
- Tester si un fichier est ouvert (en particulier si vous avez les droits de lecture sur le fichier)
- Lecture des données écrites en binaire
- Fermer le fichier une fois la lecture terminée

Voici le fichier source :

```
#include <iostream>
#include <fstream>

int main (int argc, char * argv[]) {
  std::ifstream myfile;
  char * filename = "exampleBinary.txt";
  char buffer[29];
  int value;
  myfile.open (filename, std::ios::in | std::ios::binary);
  if(myfile.is_open())
  {
```

```
        myfile.read(buffer,29*sizeof(char));
        std::cout << buffer << std::endl;
        for(int i = 0 ; i <= 1000 ; i++)
        {
            myfile.read((char*)&value,sizeof(int));
            std::cout << value << std::endl;
        }
    }
    else
    {
        std::cout << "Erreur à l'ouverture du fichier "<< filename << std::endl;
    }
    myfile.close();
    return 0;
}
```

Liens Externes

- (anglais) La page dont on s'est inspirée pour écrire les exercices (http://www.cplusplus.com/doc/tutorial/files.html) [archive]