

Travaux Pratiques

COURS



✚ Ce TP réalisé par :

ZRAIDI NAJWA
(GLSID1)

Mécanisme de surdéfinition d'opérateur :

Exemple et Exercice 1 :

```
#include<iostream>
using namespace std;
//classe vecteur
//Surdefinition de l'opérateur + et l'opérateur *(Exercice 1)

class vecteur
{
    float x,y;
public :
    vecteur(float,float);
    vecteur operator +(vecteur);
    /* surdefinition de l'opérateur somme
    on passe un paramètre un vecteur
    la fonction retourne un vecteur */
    void affiche();
    float operator*(vecteur);
    vecteur operator*(float);
};

vecteur :: vecteur(float abs=0,float ord=0)
{
    x=abs;
    y=ord;
}

void vecteur :: affiche(){
    cout<<" x = "<<x<<" y= "<<y<<"\n";
}

vecteur vecteur :: operator+(vecteur v){
    vecteur res;
    res.x=v.x+x;
    res.y=v.y+y;
    return res ;
}

float vecteur :: operator*(vecteur v){
    return v.x*x+v.y*y;
}

vecteur vecteur :: operator*(float val){
    vecteur res;

    res.x=x*val;
    res.y=y*val;
    return res ;
}

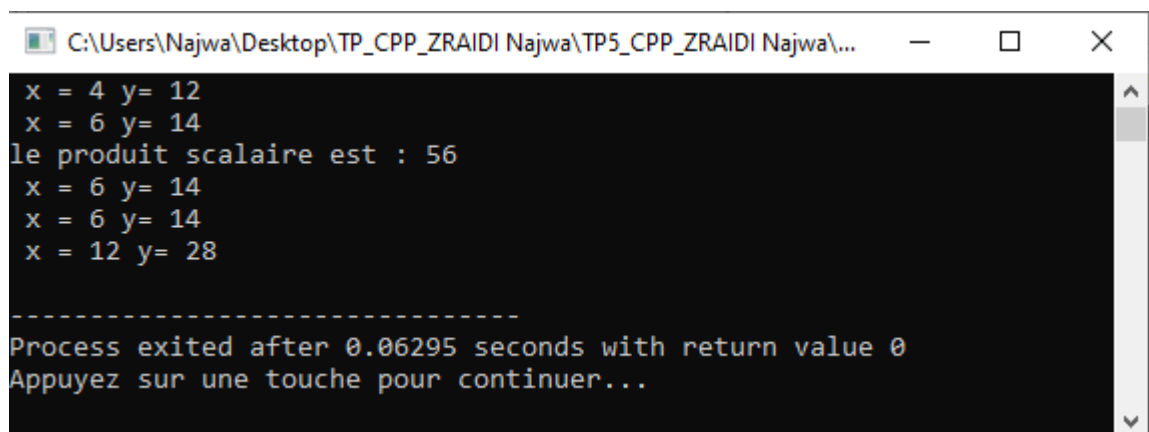
int main()
{
```

```

float prod_scal;
vecteur a(2,6),b(4,8),c,d,e,f,g;
g=a.operator*(2);
g.affiche();
c= a+b;
c.affiche();
prod_scal = a.operator*(b);
cout<< "le produit scalaire est : "<<prod_scal<<endl;
d=a.operator +(b);
d.affiche();
e=b.operator +(a);
e.affiche();
f=a+b+c;
f.affiche();
}

```

✓ L'exécution de programme donne :



```

C:\Users\Najwa\Desktop\TP_CPP_ZRAIDI Najwa\TP5_CPP_ZRAIDI Najwa\...
x = 4 y= 12
x = 6 y= 14
le produit scalaire est : 56
x = 6 y= 14
x = 6 y= 14
x = 12 y= 28

-----
Process exited after 0.06295 seconds with return value 0
Appuyez sur une touche pour continuer...

```

Application : UTILISATION DANS UNE BIBLIOTHEQUE :

APPLICATION.hpp : contient la classe complexe dont les prototypes sont déclarés .

```

#ifndef COMPLEX_H
#define COMPLEX_H
#include <istream>
#include <ostream>
using namespace std;
class complex
{
    double re,im;
public :

```

```

// partie reel et imaginaire du nombre complexe
complex(double reel=0,double imag = 0); //constructeur
//complex manipulation
double real(); //retourne la partie reel
double imag(); //retourne la partie imaginaire
complex conj(complex); //the complex conjugate
double norm(complex); //the square of the magnitude
double arg(complex); // the angle in radians
//create a complex object givin polar coordinates
complex polar(double mag,double angle=0);
//Binary Operator functions
complex operator+(complex);
friend complex operator+(double,complex);
//donnent une signification aux deus
friend complex operator+(complex,double);
//la notion de fonction amie sera etudiee lors du prochain chapitre
complex operator -(complex);
friend complex operator-(double,complex);
friend complex operator-(complex,double);
complex operator *(complex);
friend complex operator*(double,complex);
friend complex operator*(complex,double);
complex operator /(complex);
friend complex operator/(double,complex);
friend complex operator/(complex,double);
int operator==(complex);
int operator!=(complex);
complex operator-();
friend ostream& operator<<(ostream& ,const complex& );//permet d'utiliser
cout avec un complex
friend istream& operator>>(istream& , complex& );//permet d'utiliser cin
avec un complex
};

#endif

```

APPLICATION.cpp : contient l'implémentation des prototypes des méthodes qui sont déclarés dans le fichier **APPLICATION.hpp** .

```

#include <istream>
#include <ostream>
#include <iostream>
#include <math.h>
#include "Application.h"
using namespace std;
complex :: complex(double reel,double imag)//constructeur
{

```

```

    this->re=reel;
    this->im=imag;
}
//complex manipulation
double complex :: real()//retourne la partie reel
{
    return re;
}
double complex :: imag()//retourne la partie imaginaire
{
    return im;
}
complex complex :: conj(complex NC) //the complex conjugate
{
    complex res;
    res.re = NC.re;
    res.im = -NC.im;
    return res;
}
double complex :: norm(complex NC)//the square of the magnitude
{
    return sqrt((NC.re*NC.re)+(NC.im*NC.im));
}
double complex :: arg(complex NC) // the angle in radians
{
    const float PI =3.14;
    if(NC.im==0)return 0;
    else
    if(NC.re==0)
        if(NC.im>0)
            return PI/2;
        else return -PI/2;
    else
        return(atan(NC.im/NC.re));
}

//create a complex object givin polar coordinates
complex complex :: polar(double mag,double angle){
    complex res;
    res.re= mag * cos(angle);
    res.im = mag *sin(angle);
    return res;
}
complex complex :: operator+(complex NC){
    complex res;
    res.re=this->re+NC.re;
    res.im=this->im+NC.im;
    return res;
}
complex operator+(double d,complex NC){

```

```

    complex res;
    res.re=d+NC.re;
    res.im=NC.im;
    return res;
} //donnent une signification aux deus
complex operator+(complex NC,double d){
    complex res;
    res.re=NC.re+d;
    res.im=NC.im;
    return res;
}
complex complex :: operator-(complex NC){
    complex res;
    res.re=this->re-NC.re;
    res.im=this->im-NC.im;
    return res;
}
complex operator-(double d,complex NC){
    complex res;
    res.re=d-NC.re;
    res.im=NC.im-d;
    return res;
}
complex operator-(complex NC,double d){
    complex res;
    res.re=NC.re-d;
    res.im=NC.im;
    return res;
}
complex complex :: operator*(complex NC){
    complex res;
    res.re=this->re*NC.re;
    res.im=this->im*NC.im;
    return res;
}
complex operator*(double d,complex NC){
    complex res;
    res.re=d*NC.re;
    res.im=d*NC.im;
    return res;
}
complex operator*(complex NC,double d){
    complex res;
    res.re=NC.re*d;
    res.im=NC.im*d;
    return res;
}
complex complex :: operator/(complex NC){
    complex res,num,denum;

```

```

        num=(*this).operator *(NC.conj(NC));
        denum=NC.operator *(NC.conj(NC));
        res.im=num.im/denum.re;
        res.re=num.re/denum.re;
        return res;
    }
    complex operator/(double d,complex NC){
        complex res;
        res.re=d/NC.re;
        res.im=d/NC.im;
        return res;
    }
    complex operator/(complex NC,double d){
        complex res;
        if(d!=0){
            res.re=NC.re/d;
            res.im=NC.im/d;
            return res;
        }
        return -1;
    }
    int complex :: operator==(complex NC){
        if(this->re == NC.re && this->im==NC.im)
            return 1;
        else
            return 0;
    }
    int complex :: operator!=(complex NC){
        if(this->re != NC.re || this->im!=NC.im)
            return 1;
        else
            return 0;
    }
    ostream& operator<<(ostream& printOut,complex& NC){
        printOut<<NC.real()<< "+" <<NC.imag()<<"i";
        return printOut;
    }
    istream& operator>>(istream& printIn, complex& c){
        printIn >> c.re >> c.im;
        return printIn;
    }
    int main(){
        complex i,s,C,p,sum,prod,sus,div;
        int choix;
        double d;
        cout<<"-----\n";
        cout<<"L'AFFICHAGE ET LA LECTEUR DES NOMBRES COMPLEXE : "<<endl;
        cout<<"-----\n";
        cout<<"\n Entrer un nombre complexe i est : "<<endl;
    }

```

```

operator>>(cin,i);
cout<<" Le nombre complexe i est : "<<endl;
operator<<(cout,i);
cout<<"\n Entrer un nombre complexe S est : "<<endl;
operator>>(cin,s);
cout<<" Le nombre complexe S est : "<<endl;
operator<<(cout,s);
cout<<"\n-----\n";
cout<<"LES OPERATIONS SUR LES NOMBRES COMPLEXE "<<endl;
cout<<"-----\n";
cout<<"
                        Menu : " <<endl;
cout<<" 1- LE CONJUGASON DU NOMBRE COMPLEXE "<<endl;
cout<<" 2- LE MODULE DU NOMBRE COMPLEXE "<<endl;
cout<<" 3- L'ARGUMENT DU NOMBRE COMPLEXE "<<endl;
cout<<" 4- LE POLAR DU NOMBRE COMPLEXE "<<endl;
cout<<" 5- LA SOMME "<<endl;
cout<<" 6- LE PRODUIT "<<endl;
cout<<" 7- LA SUSTRACTION "<<endl;
cout<<" 8- LA DIVISION "<<endl;
cout<<" 9- L'EGALITE DE DEUX NOMBRES COMPLEXES equal()<<endl;
cout<<" 10- L'EGALITE DE DEUX NOMBRES COMPLEXES not equal()<<endl;
cout<<"-----\n";
cout<<"Entrer votre choix : \n ";
cin>>choix;
switch(choix){
case 1:
C=s.conj(s);
cout<<"\n Le conjugason de i est : "<<endl;
operator<<(cout,C);
C=i.conj(i);
cout<<"\n Le conjugason de S est : "<<endl;
operator<<(cout,C);
break;
case 2:
cout<<"\n Le norm de S est : "<<endl;
cout<<s.norm(s)<<endl;
cout<<"\n Le norm de i est : "<<endl;
cout<<i.norm(i)<<endl;
break;
case 3:
cout<<" L'argument de S est : "<<endl;
cout<<s.arg(s)<<endl;
cout<<" L'argument de i est : "<<endl;
cout<<i.arg(i)<<endl;
break;
case 4:
cout<<" Le polar de S est : "<<endl;
p=i.polar(2,10);
operator<<(cout,p);

```



```

cout<<" Le polar de S est : "<<endl;
p=s.polar(2,10);
operator<<(cout,p);
break;
case 5:
cout<<"\n-----\n";
cout<<"
                LA SOMME : "<<endl;
cout<<"-----\n";
cout<<"\n La somme de nombre complexe S et le nombre i est : "<<endl;
sum=i.operator+(s);
operator<<(cout,sum);
cout<<"\nEntrer le nombre double : "<<endl;
cin>>d;
cout<<"\n La somme de nombre complexe i et le nombre double est : "<<endl;

sum=operator+(i,d);
operator<<(cout,sum);
cout<<"\n La somme de nombre double et le nombre complexe i est : "<<endl;
sum=operator+(d,i);
operator<<(cout,sum);
break;
case 6 :
cout<<"\n-----\n";
cout<<"
                LE PRODUIT : "<<endl;
cout<<"-----\n";
cout<<"\n Le produit de nombre complexe S et le nombre i est : "<<endl;
prod=i.operator*(s);
operator<<(cout,prod);
cout<<"\nEntrer le nombre double : "<<endl;
cin>>d;
cout<<"\n Le produit de nombre complexe i et le nombre double est :
"<<endl;
prod=operator*(i,d);
operator<<(cout,prod);
cout<<"\n Le produit de nombre double et le nombre complexe i est :
"<<endl;
prod=operator*(d,i);
operator<<(cout,prod);
break;
case 7:
cout<<"\n-----\n";
cout<<"
                LA SUSTRACTION : "<<endl;
cout<<"-----\n";
cout<<"\n La sustraction de nombre complexe S et le nombre i est :
"<<endl;
sus=i.operator-(s);
operator<<(cout,sus);
cout<<"\nEntrer le nombre double : "<<endl;
cin>>d;

```

```

    cout<<"\n La sustraction de nombre complexe i et le nombre double est :
"<<endl;
    sus=operator-(i,d);
    operator<<(cout,sus);
    cout<<"\n La sustraction de nombre double et le nombre complexe i est :
"<<endl;
    sus=operator-(d,i);
    operator<<(cout,sus);
    break;
    case 8:
    cout<<"\n-----\n";
    cout<<"          LA DIVISION : "<<endl;
    cout<<"-----\n";
    cout<<"\n La division de nombre complexe S et le nombre i est : "<<endl;
    div=i.operator/(s);
    operator<<(cout,div);
    cout<<"\nEntrer le nombre double : "<<endl;
    cin>>d;
    cout<<"\n La division de nombre complexe i et le nombre double est :
"<<endl;
    div=operator/(i,d);
    if(div==1){cout<<"Impossible de diviser sur 0 "<<endl; }
    else operator<<(cout,div);
    cout<<"\n La division de nombre double et le nombre complexe i est :
"<<endl;
    div=operator/(d,i);
    operator<<(cout,div);
    break;
    case 9:
    cout<<"\n-----\n";
    cout<<"    L'egalite de nombre complexe S et le nombre i    : "<<endl;
    if(i.operator==(s))
    cout<<"\n Le nombre complexe S = Le nombre complexe i ";
    cout<<"-----\n";
    break;
    case 10:
    cout<<"\n-----\n";
    cout<<"    L'egalite de nombre complexe S et le nombre i    : "<<endl;
    if(i.operator!=(s))
    cout<<"\n Le nombre complexe S != Le nombre complexe i ";
    break;
    }
}

```

✓ L'exécution de programme donne :

C:\Users\Najwa\Desktop\TP_CPP_ZRAIDI Najwa\TP5_CPP_ZRAIDI Najwa\Cours\Application.exe

```
-----
L'AFFICHAGE ET LA LECTEUR DES NOMBRES COMPLEXE :
-----

  Entrer un nombre complexe i est :
4
2
  Le nombre complexe i est :
4+2i
  Entrer un nombre complexe S est :
1
1.5
  Le nombre complexe S est :
1+1.5i
-----
LES OPERATIONS SUR LES NOMBRES COMPLEXE
-----
                        Menu :
1- LE CONJUGASON DU NOMBRE COMPLEXE
2- LE MODULE DU NOMBRE COMPLEXE
3- L'ARGUMENT DU NOMBRE COMPLEXE
4- LE POLAR DU NOMBRE COMPLEXE
5- LA SOMME
6- LE PRODUIT
7- LA SUSTRACTION
8- LA DIVISION
9- L'EGALITE DE DEUX NOMBRES COMPLEXES equal()
10- L'EGALITE DE DEUX NOMBRES COMPLEXES not equal()
-----
Entrer votre choix :
5
-----
                        LA SOMME :
-----

  La somme de nombre complexe S et le nombre i est :
5+3.5i
  Entrer le nombre double :
7

  La somme de nombre complexe i et le nombre double est :
11+2i
  La somme de nombre double et le nombre complexe i est :
11+2i
-----
Process exited after 10.25 seconds with return value 0
Appuyez sur une touche pour continuer...
```

Les possibilités et limites de la surdéfinition d'opérateurs en C++:

Exemple :

```
#include<iostream>
#include<conio.h>
using namespace std;
class liste{
    int taille;
    float *adr;
public:
    liste(int); //constructeur
    liste(liste &); //constructeur par recopie
    void saisie();
    void affiche();
    void operator=(liste &); //surdefinition de l'operateur =
    // ~liste();
};
liste::liste(int t)
{
    taille = t; adr=new float[taille];
    cout<<"Construction \n";
    cout<<" Adresse de l'objet: "<<this;
    cout<<" Adresse de liste: "<<adr<<"\n";
}
liste::liste(liste &v)
{
    taille = v.taille; adr = new float[taille];
    for(int i=0; i<taille; i++)
        adr[i] = v.adr[i];
    cout<<"\nConstructeur par recopie";
    cout<<" Adresse de l'objet:"<<this;
    cout<<" Adresse de liste:"<<adr<<"\n";
}
void liste::saisie()
{
    int i;
    for(i=0; i<taille; i++){
        cout<<" Entrer un nombre: ";
        cin>>*(adr+i);
    }
}
void liste::affiche()
{
    int i;
    cout<<" Adresse: "<<this<<" ";
```

```

        for(i=0;i<taille;i++)
            cout<<*(adr+i)<<" ";
        cout<<"\n\n";}
void liste::operator=(liste &lis)
/*passage par reference
pour eviter l'appel au constructeur
et la double liberation d'un meme
emplacement memoire */
{   int i;
    taille=lis.taille;
    delete adr;
    adr=new float[taille];
    for(i=0;i<taille;i++)
        adr[i] = lis.adr[i];}
int main ()
{ cout<<"Debutde main()\n";
    liste a(5);
    liste b(2);
    a.saisie();
    a.affiche();
    b.saisie();
    b.affiche();
    b=a;
    b.affiche();
    a.affiche();
    cout<<"Fin de main() \n";}

```

✓ L'exécution de programme donne :

```

C:\Users\Najwa\Desktop\TP_CPP_ZRAIDI Najwa\TP5_CPP_ZRAIDI Najwa\Cours\Exe...
Debutde main()
Construction
  Adresse de l'objet: 0x6ffe10  Adresse de liste: 0x1d15f0
Construction
  Adresse de l'objet: 0x6ffe00  Adresse de liste: 0x1d1610
Entrer un nombre: 10
Entrer un nombre: 4
Entrer un nombre: 2
Entrer un nombre: 4
Entrer un nombre: 7
Adresse: 0x6ffe10 10 4 2 4 7

Entrer un nombre: 2
Entrer un nombre: 1
Adresse: 0x6ffe00 2 1

Adresse: 0x6ffe00 10 4 2 4 7

Adresse: 0x6ffe10 10 4 2 4 7

Fin de main()

-----
Process exited after 12.2 seconds with return value 0
Appuyez sur une touche pour continuer...

```

Exercice 1 :

```
#include<iostream>
#include<conio.h>
using namespace std;
class liste{
    int taille;
    float *adr;
public:
    liste(int); //constructeur
    liste(liste &); //constructeur par recopie
    void saisie();
    void affiche();
    void operator=(liste &); //surdefinition de l'operateur =
    float &operator[](int);
    // ~liste();
};
liste::liste(int t)
{
    taille = t; adr=new float[taille];
    cout<<"Construction \n";
    cout<<" Adresse de l'objet: "<<this;
    cout<<" Adresse de liste: "<<adr<<"\n";
}
liste::liste(liste &v)
{
    taille = v.taille; adr = new float[taille];
    for(int i=0; i<taille; i++)
        adr[i] = v.adr[i];
    cout<<"\nConstructeur par recopie";
    cout<<" Adresse de l'objet:"<<this;
    cout<<" Adresse de liste:"<<adr<<"\n";
}
void liste::saisie()
{
    int i;
    for(i=0; i<taille; i++){
        cout<<" Entrer un nombre: ";
        cin>>*(adr+i);
    }
}
void liste::affiche()
{
    int i;
```

```

        cout<<" Adresse: "<<this<<" ";
        for(i=0;i<taille;i++)
            cout<<*(adr+i)<<" ";
        cout<<"\n\n";
    }
    void liste::operator=(liste &lis)
    /*passage par reference
    pour eviter l'appel au constructeur
    et la double liberation d'un meme
    emplacement memoire */
    {
        int i;
        taille=lis.taille;
        delete adr;
        adr=new float[taille];
        for(i=0;i<taille;i++)
            adr[i] = lis.adr[i];
    }
    float &liste::operator[](int i)
    {
        adr[i]=this->adr[i];
    }
    int main ()
    {
        cout<<"Debutde main()\n";
        liste a(5);
        liste b(2);
        a.saisie();
        a.affiche();
        b.saisie();
        b.affiche();
        b=a;
        b.affiche();
        a.affiche();
        cout<<"Fin de main() \n";
    }

```

✓ L'exécution de programme donne :

```
C:\Users\Najwa\Desktop\TP_CPP_ZRAIDI Najwa\TP5_CPP_ZRAIDI Najwa\Cours\exercice .exe
Debutde main()
Construction
  Adresse de l'objet: 0x6ffe10  Adresse de liste: 0xa91570
Construction
  Adresse de l'objet: 0x6ffe00  Adresse de liste: 0xa91590
Entrer un nombre: 4
Entrer un nombre: 2
Entrer un nombre: 1
Entrer un nombre: 4
Entrer un nombre: 7
Adresse: 0x6ffe10 4 2 1 4 7

Entrer un nombre: 4
Entrer un nombre: 4
Adresse: 0x6ffe00 4 4

Adresse: 0x6ffe00 4 2 1 4 7

Adresse: 0x6ffe10 4 2 1 4 7

Fin de main()

-----
Process exited after 7.479 seconds with return value 0
Appuyez sur une touche pour continuer...
```

Exercice 2 :

```
#include <string.h>
#include<iostream>
#include<conio.h>
using namespace std;

class String
{
private:
    char* string_ptr;    //Pointeur sur le texte.
    long size;           //Nombre de caractères du texte (sans compter le
caractère final NULL.)
public:
    String();
    String(const char* chaine); //Constructeur basé sur un argument de type
chaîne de caractères.
    String(const String& chaine); //Constructeur de copie d'une autre
instance String.
    String operator+(char* chaine); //Opérateur + pour la concaténation avec
une chaîne.
    String operator+(const String& chaine); //Opérateur + pour la
concaténation avec une instance String.
```



```

    friend String operator+(char* str,const String& chaine); //Opérateur+ pour
la concaténation entre une chaîne et une instance String.
    String operator=(const char* chaine); //Opérateur = pour l'attribution
d'une chaîne dans une instance String.
    String operator=(const String& chaine); //Opérateur = pour la copie
d'une instance String.
    char* GetStringPtr(); //Permet de récupérer le pointeur texte.
    long GetStringLen(); //Permet de récupérer le nombre de caractères du
texte.
    ~String(); //Destructeur permettant de libérer la place empli
par les instances destinées à être détruites.
};

```

```
String::String()
```

```
{
    string_ptr = NULL;
    size = 0;
}
```

```
String::String(const char* chaine)
```

```
{
    long Size = strlen(chaine);
    size = Size;
    char* ptr = new char[size+1];
    ptr[size]=0;
    strcpy(ptr,chaine);
    string_ptr = ptr;
}
```

```
String::String(const String& chaine)
```

```
{
long Size = strlen(chaine.string_ptr);
size = Size;
char* ptr = new char[size+1];
ptr[size]=0;
strcpy(ptr,chaine.string_ptr);
string_ptr = ptr;
}
```

```
String String::operator+(char* chaine)
```

```
{
String Copie;
long NewLen = (this->size+strlen(chaine));

```

```

Copie.string_ptr = new char[NewLen+1];
Copie.size = NewLen;
Copie.string_ptr[NewLen]=0;
strcpy(Copie.string_ptr,this->string_ptr);
strcat(Copie.string_ptr,chaine);
return Copie;
}

String String::operator+(const String& chaine)
{
String Copie;
long NewLen = (this->size+chaine.size);
Copie.string_ptr = new char[NewLen+1];
Copie.size = NewLen;
Copie.string_ptr[NewLen]=0;
strcpy(Copie.string_ptr,this->string_ptr);
strcat(Copie.string_ptr,chaine.string_ptr);
return Copie;
}

String operator+(char* str,const String& chaine)
{
String Copie;
long NewLen = (strlen(str)+chaine.size);
Copie.string_ptr = new char[NewLen+1];
Copie.size = NewLen;
Copie.string_ptr[NewLen]=0;
strcpy(Copie.string_ptr,str);
strcat(Copie.string_ptr,chaine.string_ptr);
return Copie;
}

String String::operator=(const char* chaine)
{
this->size = strlen(chaine)+1;
char* ptr = new char[this->size];

strcpy(ptr,chaine);
delete[] this->string_ptr;
this->string_ptr = ptr;
this->string_ptr[(this->size)-1]=0;
return *this;
}

String String::operator=(const String& chaine)

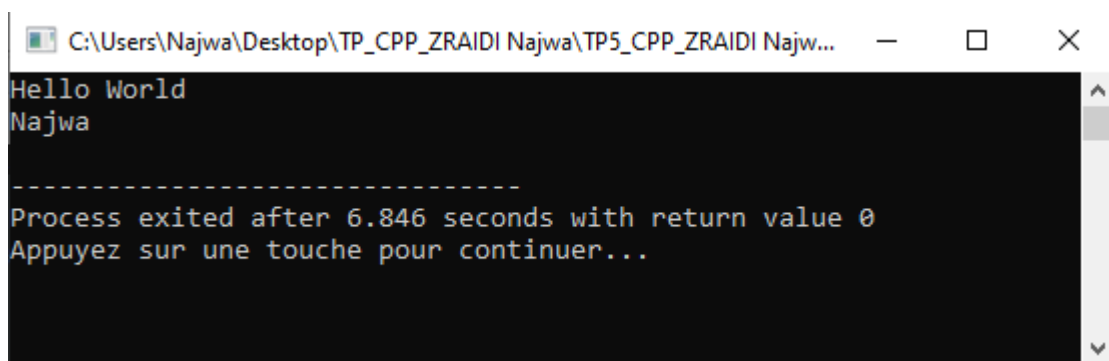
```

```

{
this->size = strlen(chaine.string_ptr)+1;
char* ptr = new char[this->size];
strcpy(ptr, chaine.string_ptr);
delete[] this->string_ptr;
this->string_ptr = ptr;
this->string_ptr[(this->size)-1]=0;
return *this;
}
char* String::GetStringPtr()
{
return string_ptr;
}
long String::GetStringLen()
{
return size;
}
String::~~String()
{
delete []string_ptr;
string_ptr = NULL;
size = 0;
}
int main(int argc, char *argv[])
{
String A = "Hello";
String B = " World";
A=A+B;
printf("%s\n", A.GetStringPtr());
getchar();
}

```

✓ L'exécution de programme donne :



```

C:\Users\Najwa\Desktop\TP_CPP_ZRAIDI Najwa\TP5_CPP_ZRAIDI Najw...
Hello World
Najwa

-----
Process exited after 6.846 seconds with return value 0
Appuyez sur une touche pour continuer...

```