14/20

CannizzaroLucas_SchatzGregory_IPV6.cpp

Fonctionne avec les adresses

IPV6 ip1("2001:0db8:0000:85a3:0000:0000:ac1f:0001");
IPV6 ip4("0000:0000:0000:0000:0000:0000:0000:0001");
IPV6 ip5("0000:0000:0000:1234:6678:0000:0000:0001");

Plantage sur les adresses suivantes :

IPV6 ip2("2001:0db8:0000:0000:0000:0000:ac1f:0000");
IPV6 ip3("2001:0db8:0000:85a3:64BC:0000:ac1f:0000");

+ 5 points sur 20 si vous corrigez votre code

```
/**********************
IPV6.h
**********************/
#pragma once
#include <string>
#include <vector>

using namespace std;

class IPV6
{
private:
        string adresse;
        vector<string> strHextets;
        bool error;
        int position;
        int memo;

        void suppressionDesZerosDeDebut();
        void remplacementSuiteDeZeros();
        void reconstructionFinale();

        static vector<string> split(const string& s, char c);

public:
        IPV6(string adr);

        string getAdresse()
        {
                reconstructionFinale();
                return adresse;
        }


        bool getError()
        {
                return error;
        }
};



/**********************
IPV6.cpp
**********************/
#include <vector>
#include <string>
#include "IPV6.h"

using namespace std;

IPV6::IPV6(string adr)
{
        this->adresse = adr;
        this->strHextets = IPV6::split(this->adresse, ':');
        this->error = false;
        if (this->strHextets.size() == 8) {
                for (unsigned i = 0; i < 8; i++) {
                        if (this->strHextets[i].length() != 4) {
                                this->error = true;
                        }
                }
        }
        else {
                this->error = true;
        }
}

void IPV6::suppressionDesZerosDeDebut()
{
        for (unsigned int i = 0; i < this->strHextets.size(); i++)
        {
                for (unsigned int j = 0; j < this->strHextets[i].length(); j++) {
                        int flag = 0;
                        while (flag == 0) {
                                if (this->strHextets[i][0] == '0' && j < this->strHextets[i].length() - 1) {
                                        this->strHextets[i].erase(0, 1);
                                }
                                else {
                                        flag = 1;
                                }
                        }
                }
        }
}
void IPV6::remplacementSuiteDeZeros()
```

Le contenu de cette méthode est très lourd ! Ne la codez pas inline !

OK

OK

```
{
	int flag = 0, postmp = 0, memotmp = 0, j;
	for (unsigned int i = 0; i < this->strHextets.size(); i++)
	{
		if (this->strHextets[i] == "0" && flag == 0) {
			flag = 1;
			this->position = i;
			j = i;
			while (this->strHextets[j] == "0" && j < this->strHextets.size()) {
				this->memo++;
				j++;
			}
			i = i + (j - i);
		}
		else if (this->strHextets[i] == "0" && flag == 1) {
			postmp = i;
			j = i;
			while (this->strHextets[j] == "0" && j < this->strHextets.size()) {
				memotmp++;
				j++;
			}
			i = i + (j - i);
		}

		if (this->memo < memotmp) {
			this->memo = memotmp;
			this->position = postmp;
			memotmp = 0;
		}
		if (this->memo == memotmp && postmp > this->position) {
			this->position = postmp;
			memotmp = 0;
			postmp = 0;
		}
		else {
			memotmp = 0;
			postmp = 0;
		}
	}
	this->strHextets.erase(this->strHextets.begin() + (this->position + 1), this->strHextets.begin() + (this
->position + this->memo));
	this->strHextets.at(this->position) = ":";
}
void IPV6::reconstructionFinale()
{
	suppressionDesZerosDeDebut();
	remplacementSuiteDeZeros();
	this->adresse = this->strHextets[0];
	for (unsigned int i = 1; i < this->strHextets.size(); i++)
	{
		if (i != this->position) {
			this->adresse += ":" + this->strHextets[i];
		}
		else {
			this->adresse += this->strHextets[i];
		}
	}
}

vector<string> IPV6::split(const string & s, char c)
{
	vector<string> v;
	unsigned int i = 0;
	unsigned int j = s.find(c);
	while (j < s.length()) {
		v.push_back(s.substr(i, j - i));
		i = ++j;
		j = s.find(c, j);
		if (j >= s.length()) {
			v.push_back(s.substr(i, s.length()));
			break;
		}
	}
	return v;
}

/**********************
main.cpp
**********************/
#include <iostream>
#include <vector>
#include <string>
#include <conio.h>
#include "IPV6.h"

using namespace std;
```

C'est une usine à gaz !
Vous n'avez écrit aucun commentaire, donc c'est quasi impossible pour moi de comprendre et corriger votre code !

```
int main(void)
{
        string adr;
        vector<string> hext;
/*      bool error;
        do {
                cout << "Veuillez taper une adresse IPV6 : ";
                getline(cin, adr);
                IPV6 adr1(adr);
                error = adr1.getError();
                if (error) {
                        cout << "\nAdresse IPV6 incorrect !!!\nVeuillez une donner une autre !" << endl;
                }
        } while (error);
        IPV6 adr1(adr);
        cout << "L'adresse IPV6 compresse est : " << adr1.getAdresse() << endl;

*/

        IPV6 ip1("2001:0db8:0000:85a3:0000:0000:ac1f:0001");
        cout << ip1.getAdresse() << endl;

        IPV6 ip2("2001:0db8:0000:0000:0000:0000:ac1f:0000");
        cout << ip2.getAdresse() << endl;

        IPV6 ip3("2001:0db8:0000:85a3:64BC:0000:ac1f:0000");
        cout << ip3.getAdresse() << endl;

        IPV6 ip4("0000:0000:0000:0000:0000:0000:0000:0001");
        cout << ip4.getAdresse() << endl;

        IPV6 ip5("0000:0000:0000:1234:6678:0000:0000:0001");
        cout << ip5.getAdresse() << endl;




        cin.get();
        cin.ignore();
        return EXIT_SUCCESS;
}
```