Compte rendu TD 1 C++

Maxime Soulie

1

```cpp
#include <iostream>

using namespace std;

int main(int argc, char *argv[]){
    cout << "Premier exercice c++ en cours ENSIBS 3 ème année" << endl;
    std::cout << "Nombre de paramètres : " << argc - 1 << std::endl;
}
```

2

```cpp
#include <iostream>

using namespace std;

int main(int argc, char *argv[]){
        //attends un entier en entree et affiche sa valeur + 1
    cout << "entrez un nombre : ";
    int value;
    cin >> value;
    cout << value+1;

        //attends 1 entier, un double et un tableau de char, et les
affiche en sortie dans l'ordre
    int valInt;
    double valDouble;
    string valString;
    cout << endl << "entrez un entier, un double et un str : ";

    cin >> valInt >> valDouble >> valString;
    cout << valInt << " " << valDouble << " " << valString << endl;

    return 0;
}
```

trace d'execution :

```
max@archlinux ~/C/3/c/T/output (main)> ./"2"
entrez un nombre : 1
2
entrez un entier, un double et un str : 1 2.5 ensibs
1 2.5 ensibs
max@archlinux ~/C/3/c/T/output (main)> ./"2"
entrez un nombre : 1
2
entrez un entier, un double et un str : 2.5 1 ensibs
2 0.5 1
```

Pour la 2e execution : le programme prend en compte la partie entiere du 2.5, puis la partie decimale du 2.5, la 3e valeure est donc le 1. ensibs n'est alors pas enrgistree.

3

Dans un double : 18 decimales de pi (19 max dont 1 pour la partie entiere)

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

int main() {
// affiche la taille de la memoire allouee pour chaque type
    cout << "Taille de int : " << sizeof(int) << " octets" << endl
    << "Taille de short : " << sizeof(short) << " octets" << endl
    << "Taille de float : " << sizeof(float) << " octets" << endl
    << "Taille de double : " << sizeof(double) << " octets" << endl
    << "Taille de long double : " << sizeof(long double) << " octets" <<
endl;

//permet de comparer la precision d'un long double
    long double pi =
3.14159265358979323846264338327950288419716939937510582;
    cout<< setprecision(99) << endl << "Valeur de pi : " << pi << endl<<
endl;

// affiche les limites des int
    signed int INT_MIN = numeric_limits<int>::min();
    signed int INT_MIN_MOINS1 = numeric_limits<int>::min()-1;
    signed int INT_MAX = numeric_limits<int>::max();
    signed int INT_MAX_PLUS1 = numeric_limits<int>::max()+1;

    cout << "INT_MIN : " << INT_MIN << endl
    << "INT_MIN-1 : " << INT_MIN_MOINS1 << endl
    << "INT_MAX : " << INT_MAX << endl
    << "INT_MAX+1 : " << INT_MAX_PLUS1 << endl<< endl;
```

```
// affiche les limites des long double
    long double LDBL_MIN = numeric_limits<long double>::min();
    long double LDBL_MIN_MOINS1 = numeric_limits<long double>::min()-1;
    long double LDBL_MAX = numeric_limits<long double>::max();
    long double LDBL_MAX_PLUS1 = numeric_limits<long double>::max()+1;

    cout << "LDBL_MIN : " << LDBL_MIN << endl
    << "LDBL_MIN-1 : " << LDBL_MIN_MOINS1 << endl
    << "LDBL_MAX : " << LDBL_MAX << endl
    << "LDBL_MAX+1 : " << LDBL_MAX_PLUS1 << endl;

    return 0;
}
```

trace d'execution :

```
Taille de int : 4 octets
Taille de short : 2 octets
Taille de float : 4 octets
Taille de double : 8 octets
Taille de long double : 16 octets

Valeur de pi : 3.14159265358979311599796346854418516159057617187 5

INT_MIN : -2147483648
INT_MIN-1 : 2147483647
INT_MAX : 2147483647
INT_MAX+1 : -2147483648

LDBL_MIN :
3.3621031431120935062626778173217526025980793448464712401088272298087426993907289670430927063650562 2e-4932
LDBL_MIN-1 : -1
LDBL_MAX :
1.1897314953572317650212638530309702051690633222946242004403237338917370055229707226164102903365288 8e+4932
LDBL_MAX+1 :
1.1897314953572317650212638530309702051690633222946242004403237338917370055229707226164102903365288 8e+4932
```

On peut voir apres la 15e decimale, le nombre entré et le nombre sortie different.

La valeur max d'un int + 1

4

```cpp
#include <iostream>
#include<map>
#include<string>
using namespace std;

int main(int argc, char *argv[]){
    enum Week  {LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI, SAMEDI,
DIMANCHE};
    Week day;
    int value;
    cout << "entrez un nombre entre 1 et 7 : ";
    cin >> value;
    day = static_cast<Week>(value-1);
    switch (day)
    {
        case LUNDI:
            cout << "LUNDI" << endl;
            break;
        case MARDI:
            cout << "MARDI" << endl;
            break;
        case MERCREDI:
            cout << "MERCREDI" << endl;
            break;
        case JEUDI:
            cout << "JEUDI" << endl;
            break;
        case VENDREDI:
            cout << "VENDREDI" << endl;
            break;
        case SAMEDI:
            cout << "SAMEDI" << endl;
            break;
        case DIMANCHE:
            cout << "DIMANCHE" << endl;
            break;
        default:
            cout << "ERROR" << endl;
            break;
    }
    return 0;
}
```

5

```cpp
#include<iostream>
#include<vector>
#include<numeric>
```

```cpp
#include<algorithm>

using namespace std;

int main(int argc, char *argv[]){
    vector<double> v;
    double values;
    cout << "entrez plusieurs doubles (entrez un char pour arreter) : ";
    while (cin >> values){
        v.push_back(values);
    }

    //somme
    cout << "somme des valeurs" << accumulate(v.begin(), v.end(), 0.0)<<
endl;

    //affichage
    cout << "affichage des valeurs en sens inverse" << endl;
    for (auto it = v.rbegin(); it != v.rend(); ++it) {
        cout << *it << " ";
    }
    cout << endl<< endl;;
    //tri
    cout << "affichage des valeurs triées : " << endl;
    sort(v.begin(), v.end());
    for(double val : v){
        cout << val << ", ";
    }

    cout << endl;

    //produit matriciel
    cout << "produit matriciel" << endl;
    vector<vector<double>> v1 {vector<double> {1,2,3}, vector<double>
{4,5,6}};
    vector<vector<double>> v2{vector<double> {1,2}, vector<double> {3,4},
vector<double> {5,6}};

    vector<vector<double>> result(v1.size(), vector<double>(v2[0].size(),
0));

        // calcul de v1xv2
    for (size_t i = 0; i < v1.size(); ++i) {
        for (size_t j = 0; j < v2[0].size(); ++j) {
            for (size_t k = 0; k < v1[0].size(); ++k) {
                result[i][j] += v1[i][k] * v2[k][j];
            }
        }
    }
```

```
    // affichage de v1xv2
    for (const auto& row : result) {
        for (const auto& element : row) {
            cout << element << " ";
        }
        cout << endl;
    }

    return 0;
}
```

trace d'execution :

```
entrez plusieurs doubles (entrez un char pour arreter) : 1 2 3 4 5 9.4 5.6
2 4 8 3 m
somme des valeurs : 47
affichage des valeurs en sens inverse
3 8 4 2 5.6 9.4 5 4 3 2 1

affichage des valeurs triées
1, 2, 2, 3, 3, 4, 4, 5, 5.6, 8, 9.4,
produit matriciel
22 28
49 64
```

calcul de v1xv2