



Solutions de l'élève Ben Strasser écrites en C++

Problème I - Paires amicales

20 points

```
#include <cstdio>
#include <algorithm>
using namespace std;

typedef unsigned uint;

// {a,a} == a
// {a,b} == {b,a}

uint div_sum[32767];

void enter_value(uint n)
{
    for(uint i = n*2; i<32767; i+=n)
        div_sum[i] += n;
}

void create_div_sum_table()
{
    fill(div_sum, div_sum+1, 0);
    for(uint i = 1; i<32767; ++i)
        enter_value(i);
}

void print_pairs(){
    for(uint a=2; a<32767; ++a){
        uint b = div_sum[a];
        if(b < 32767 && div_sum[b] == a && b != a && a < b)
            printf("%u %u\n", a, b);
    }
}

int main()
{
    create_div_sum_table();
    print_pairs();
    system("pause");
    return 0;
}
```

Problème II - Occurrences**20 points**

```

#include <cstdio>
#include <cctype>
#include <algorithm>
using namespace std;

typedef unsigned uint;
uint times[26];

int main()
{
    FILE*in = fopen("OCC_IN.TXT", "r");
    if(in)
    {
        fill(times, times+26, 0);
        for(int c=fgetc(in); c != EOF; c=fgetc(in))
            if(isalpha(c))
                times[tolower(c)-'a'] += 1;
        for(uint i=0; i<26;++i)
            printf("%c %u\n", static_cast<char>(i + 'A'), times[i]);
        fclose(in);
    }else
        fprintf(stderr, "N'a pas pu ouvrir \"OCC_IN.TXT\".\n");
    system("pause");
    return 0;
}

```

Problème III - Sudoku**30 points**

```

#include <iostream>
#include <fstream>
#include <algorithm>
#include <cassert>
using namespace std;

typedef unsigned uint;

// Board Data I/O

uint board[9][9];

void load_board(istream&in)
{
    for(uint y=0;y<9;++y){
        for(uint x=0;x<9;++x){
            if(!(in>>board[x][y])){
                cerr<<"Le fichier SUDO_IN.TXT ne contient pas assez de
nombres."<<endl;
                system("pause");
                abort();
            }
        }
    }
    uint dummy;
    if(in>>dummy)
        cerr<<"Avertissement : Le fichier SUDO_IN.TXT contient plus de
nombres que necessaire. Uniquement les 81 premiers seront consideres."<<endl;
}

```

```
void print_board(ostream&in)
{
    for(uint y=0;y<9;++y){
        for(uint x=0;x<9;++x){
            in<<board[x][y]<<' ';
        }
        in<<'\n';
    }
    in<<flush;
}

// Iterativ accessors

class Column{
public:
    Column(uint x):
        x(x){}
    uint&operator()(uint i)const{
        return board[x][i];
    }
private:
    uint x;
};

class Row{
public:
    Row(uint y):
        y(y){}
    uint&operator()(uint i)const{
        return board[i][y];
    }
private:
    uint y;
};

class Box{
public:
    Box(uint x, uint y):
        x(x*3), y(y*3){}
    uint&operator()(uint i)const{
        return board[x + i%3][y + i/3];
    }
private:
    uint x,y;
};

// generigic algorithms

template<class Acc>
bool is_one_missing(Acc seq)
{
    uint to_fill = 0;
    for(uint i=0; i<9; ++i)
        if(seq(i) == 0)
            ++to_fill;

    if(to_fill == 0)
        // All boxes are filled
        return false;
    else if(to_fill >= 2)
        // Too many boxes aren't filled
        return false;
    else
        return true;
}
```

```
}

template<class Acc>
uint find_missing_one(Acc seq)
{
    bool there[9];
    fill(there, there+9, false);
    for(uint i=0; i<9; ++i)
        if(seq(i) != 0)
            there[seq(i)-1] = true;

    for(uint i=0; i<9; ++i)
        if(!there[i])
            return i+1;
    assert(false);
}

template<class Acc>
void insert(Acc seq, uint n)
{
    for(uint i=0; i<9; ++i)
        if(seq(i) == 0)
            seq(i) = n;
}

template<class Acc>
bool complete(Acc seq)
{
    if(is_one_missing(seq)){
        uint n = find_missing_one(seq);
        insert(seq, n);
        return true;
    }
    return false;
}

// Main

void load()
{
    ifstream in("SUDO_IN.TXT");
    if(in)
        load_board(in);
    else{
        cerr<<"N'a pas pu ouvrir SUDO_IN.TXT"<<endl;
        system("pause");
        abort();
    }
}

void print()
{
    ofstream out("SUDO_OUT.TXT");
    print_board(out);
}

void complete_all()
{
    bool changed;
    do{
        changed = false;
        for(uint i = 0; i<9; ++i)
            {
                if(complete(Column(i)))
                    changed = true;
            }
    } while(changed);
}
```

```
        changed = true;
    if(complete(Row(i)))
        changed = true;
    if(complete(Box(i%3, i/3)))
        changed = true;
    }
}while(changed);
}

int main()
{
    load();
    complete_all();
    print();
    system("pause");
    return 0;
}
```

Problème IV – Maison de St. Nicolas

30 points

```
#include <fstream>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;

typedef unsigned uint;

// Connection

bool is_connected(char a, char b)
{
    if(a == b)
        return false;
    if(a != 'D' && b != 'D')
        return true;
    if(a == 'D')
        a = b;
    if(a == 'A' || a == 'B')
        return false;
    else
        return true;
}

// Path

bool went[5][5];

void reset_path()
{
    for(uint i=0; i<5; ++i)
        for(uint j=0; j<5; ++j)
            went[i][j] = false;
}

bool can_go(char a, char b)
{
    return !went[a - 'A'][b - 'A'] && is_connected(a, b);
}

void go(char a, char b)
{
    assert(can_go(a, b));
}
```

```
    assert(can_go(b, a));
    went[a - 'A'][b - 'A'] = true;
    went[b - 'A'][a - 'A'] = true;
}

void go_back(char a, char b)
{
    assert(!can_go(a, b));
    assert(!can_go(b, a));
    went[a - 'A'][b - 'A'] = false;
    went[b - 'A'][a - 'A'] = false;
}

// Back Track

void search(vector<char>&path, ostream&out, char now)
{
    if(path.size() == 9){
        copy(path.begin(), path.end(), ostream_iterator<char>(out));
        out.put('\n');
    }else{
        for(char next = 'A'; next <= 'E'; ++next){
            if(can_go(now, next))
            {
                path.push_back(next);
                go(now, next);
                search(path, out, next);
                go_back(now, next);
                path.pop_back();
            }
        }
    }
}

void search(ostream&out)
{
    vector<char>path;
    for(char start = 'A'; start <= 'E'; ++start){
        path.push_back(start);
        search(path, out, start);
        path.pop_back();
    }
}

int main()
{
    ofstream out("NICO_OUT.TXT");
    reset_path();
    search(out);
    out.flush();
    system("pause");
    return 0;
}
```