

Analyse d'algorithmes

Florian Bourse

Exemple de cours :

C

```
void div_e(int resultat[2], int a, int b) {
    //  $a = b * q + r$ ,  $resultat[0]=q$ ,  $resultat[1]=r$ 
    int q, r, n, aux;
    n = 0; aux = b;
    while (aux <= a) { //  $aux = b * 2^n$ 
        n = n + 1;
        aux = 2 * aux;
    } //  $aux = b * 2^n > a$ , le plus petit
    q = 0; r = a;
    while (n > 0) {
        /*  $aux = b * 2^n$       *
         *  $a = aux * q + r$  *
         *  $0 \leq r < aux$     *
         *  $n \geq 0$            */
        aux = aux / 2;
        n = n - 1;
        if (r < aux) {
            q = 2 * q;
        } else {
            q = 2 * q + 1;
            r = r - aux;
        }
    }
    resultat[0] = q; resultat[1] = r;
}
```

Exercices

Lorsqu'un code est donné sans consigne, il est demandé d'analyser sa terminaison à l'aide d'un variant, puis sa correction, à l'aide d'invariants. Si nécessaire, on apportera des modifications au programme pour le réparer (c'est le cas pour au moins un des programmes).

0 Un enfant très ordonné entreprend de ranger ses petits trains. Chaque train est formé d'une suite d'éléments, indistinctement wagons ou locomotives. L'enfant suit la procédure suivante :

- prendre un train non rangé
- s'il contient un seul élément, le ranger dans la caisse,
- sinon le séparer en deux trains plus petits, qui sont remis dans les trains non rangés.
- Recommencer tant qu'il reste des trains à ranger.

L'enfant ne suit aucune stratégie particulière pour choisir le prochain train à prendre, ni pour choisir l'endroit où couper un train en deux. Ce processus va-t-il s'arrêter ?

1

C

```
bool palindrome(char str[]){
    int i = 0,
        j = 0;

    while (str[j] != '\0') { // Loop 1
        j = j + 1;
    }

    j = j - 1;

    while (j >= i) { // Loop 2
        if (str[i] != str[j]) {
            return false;
        }
    }

    return true;
}
```

2

C

```
int log2(int x) {
    int l = 0;

    while (x > 0) {
        x = x / 2;
        l = l + 1;
    }

    return l;
}
```

3

C

```
int f(int x) {
    int n = 0;

    while (x > 0) {
        x = -2 * x + 10;
        n = n + 1;
    }

    return n;
}
```

4

C

```

int pgcd(int a,int b) {
    while (a != b) {
        if (a > b) {
            a = a - b;
        } else {
            b = b - a;
        }
    }

    return a;
}

```

5

C

```

int sqrt(int n) {
    int c = 0,
        s = 1;

    while (s <= n) {
        c = c + 1;
        s = s + 2 * c + 1;
    }

    return c;
}

```

6

C

```

int mystere(int p) {
    int c = 0;

    while (p > 0) {
        if (c == 0) {
            p = p - 2;
            c = 1;
        } else {
            p = p + 1;
            c = 0;
        }
    }

    return p;
}

```

7 Écrire une fonction `void swap(int* tab, int i, int j)` qui échange les valeurs de `tab[i]` et `tab[j]`.

Écrire une fonction `void insertion_sort(int* tab, int n)` pour trier un tableau `tab` de taille `n` en utilisant l'algorithme suivant : pour chaque case en partant de 0, on y met le minimum des nombres restant à trier.