

# INFO0947: Multiplicité du maximum

Groupe S190632: Luca MATAGNE,

## Table des matières

## 1 Formalisation du problème

## 2 Spécification formelle

Il est important de déterminer la précondition et la postcondition de notre problème, qui seront le point de départ (pré) et d'arrivée (post) de notre raisonnement constructif.

Voici ces conditions :

— Précondition

$$N > 0 \wedge T[N] \neq \text{NULL}$$

— Postcondition

$$\text{max} = \text{max}(T[N]) \wedge \text{multiplicite}() = \text{nombre d'occurrence du maximum}$$

## 3 Découpe en sous problèmes

Etant donné que le problème s'articule autour d'une seule boucle principale et que le fait de diviser le problème en deux SP qui seraient :

1) Recherche du maximum

2) Comptage du nombre d'occurrence du maximum

amènerait à une mécompréhension (une telle découpe laisserait croire qu'il nous faut deux boucles). Je décide de ne pas diviser le problème principal en différents SP.

## 4 Invariant graphique et formel de la boucle

Voici l'invariant graphique de ma boucle 'while' :



Et voici l'invariant formel qui en découle :

$$N > 0 \wedge T[N] \neq \text{NULL} \wedge 1 \leq i \leq N \wedge \text{max} = \text{max}(T[i]) \wedge \text{nbr} \geq 1$$

## 5 Approche constructive

Dans cette section se trouve l'approche constructive qui m'a permis de construire mon fichier "multiplicite.c"

### 5.1 Code d'initialisation de la boucle

```
1
2 //N>0 ∧ T[N] != NULL
3 int i = 1;
4 //N>0 ∧ T[N] != NULL ∧ 1≤i≤N
5 int nbr = 1;
6 //N>0 ∧ T[N] != NULL ∧ 1≤i≤N ∧ nbr≥1
7 int max = T[i];
8 //N>0 ∧ T[N] != NULL ∧ 1≤i≤N ∧ max = max(T[i]) ∧ nbr≥1
```

Extrait de Code 1 – Code d'initialisation de la boucle

### 5.2 Corps de la boucle

```
1
2 //N>0 ∧ T[N] != NULL ∧ 1≤i≤N ∧ max = max(T[i]) ∧ nbr≥1
3 while(i<N);
4 //N>0 ∧ T[N] != NULL ∧ 1≤i≤N ∧ max = max(T[i]) ∧ nbr≥1 ∧ i<N
5 if(T[i] > max){
6 //N>0 ∧ T[N] != NULL ∧ 1≤i<N ∧ max = max(T[i]) ∧ nbr≥1 ∧ i<N ∧ T[i]>max
7 max = T[i];
8 nbr = 1;
9 }
10 //N>0 ∧ T[N] != NULL ∧ 1≤i<N ∧ max = max(T[i]) ∧ nbr≥1
11 else if(T[i]==max){
12 //N>0 ∧ T[N] != NULL ∧ 1≤i<N ∧ max = max(T[i]) ∧ nbr≥1 ∧ T[i]=max
13 nbr++;
14 }
15 //N>0 ∧ T[N] != NULL ∧ 1≤i<N ∧ max = max(T[i]) ∧ nbr≥1
16 i++;
17 //N>0 ∧ T[N] != NULL ∧ 1≤i≤N ∧ max = max(T[i]) ∧ nbr≥1
```

Extrait de Code 2 – Corps de la boucle

### 5.3 Code de terminaison de la boucle

```
1
2 //N>0 ∧ T[N] != NULL ∧ max = max(T[i]) ∧ nbr≥1 ∧ i=N
3 //N>0 ∧ T[N] != NULL ∧ max = max(T[N]) ∧ nbr≥1
4 return nbr;
5 //N>0 ∧ T[N] != NULL ∧ max = max(T[N]) ∧ multiplicite = nombre d'occurrence du maximum
```

Extrait de Code 3 – Code de terminaison de la boucle

## 6 Complexité

## 7 Code complet

### 7.1 main.c

```
1 #include <stdio.h>
2 #include "multiplicite.h"
3
4
5 int main(){
6
7     int T[8] = {13, -1, 16, 9, -12, 2, 4, 16};
8     int max;
9
10    multiplicite(T, 8, &max);
11
12    printf("%d - %d\n", multiplicite(T, 8, &max), max);
13 }//fin main
```

Extrait de Code 4 – Main.c

### 7.2 multiplicite.c

```
1 #include "multiplicite.h"
2
3 int multiplicite(int *T, const int N, int *max){
4
5     int i = 1;
6     int nbr = 1;
7     *max = T[0]; // le minimum pour un entier
8
9     while(i<N){
10
11         if(T[i]> *max){
12
13             *max = T[i];
14             nbr = 1;
15
16         }else if(T[i]== *max){
17
18             nbr++;
19
20         }
21
22         i++;
23
24     }
25
26     return nbr;
27
28 }
```

Extrait de Code 5 – multiplicite.c

### 7.3 multiplicite.h

```

1 #ifndef __MULTIPLICITE_T__
2 #define __MULTIPLICITE_T__
3
4 /**
5  * multiplicite
6  *
7  * @pre : N > 0 $\land$ T[N] != NULL
8  *
9  * @post : max = max(T[N])
10 *
11 * @return : multiplicite()=nombre d'occurence du maximum
12 *
13 */
14
15 int multiplicite(int *T, const int N, int *max);
16
17 #endif

```

Extrait de Code 6 – multiplicite.h

## 7.4 makefile

```

1 CC=gcc
2 LD=gcc
3 CFLAGS=--std=c99 --pedantic -Wall -W -Wmissing-prototypes
4 LDFLAGS=
5 EXEC=output
6
7 all:$(EXEC)
8
9 output: main.o multiplicite.o
10     $(LD) -o output main.o multiplicite.o $(LDFLAGS)
11
12 main.o: main.c
13     $(CC) -c main.c -o main.o $(CFLAGS)
14
15 multiplicite.o: multiplicite.c multiplicite.h
16     $(CC) -c multiplicite.c -o multiplicite.o $(CFLAGS)

```

Extrait de Code 7 – makefile