TP 5: Les tableaux

Guillaume Aubian

Ce document a été grandement inspiré par – et largement copié sur – le document correspondant du responsable précédent de ce cours, Juliusz Chroboczek, avec son accord.

1 Préliminaires

1. Écrivez une fonction `void print_array(int a[], int n)` qui prend en paramètre un tableau d'entiers et sa taille, et affiche tous ses éléments, séparés par des espaces, puis va à la ligne.
2. Écrivez une fonction `void read_array(int a[], int n)` qui prend en paramètre un tableau et sa taille et remplit ce premier avec des entiers lus au clavier.

Testez ces deux fonctions dans main. Dans la suite de ce TP, vous devez tester toutes les fonctions que vous écrivez.

2 Maxima

1. Écrivez une fonction `int max(int a[], int n)` qui renvoie le plus grand élément d'un tableau d'entiers.
2. Écrivez une fonction `int max_pos(int a[], int n)` qui renvoie un indice du plus grand élément du tableau d'entiers passé en paramètre. Que fait votre fonction si le tableau contient plusieurs occurrences du plus grand élément ?

3 Crible d'Ératosthène

– Écrivez une fonction `int prime(int n)` qui retourne 1 si n est premier, 0 sinon (vous pouvez faire un copier-coller du TP 2).

– Écrivez un programme qui lit un entier n et affiche le nombre de nombres premiers entre 2 et n (au sens large).

Ératosthène écrit les entiers de 2 à 100 dans la cour de la Bibliothèque d'Alexandrie. Il voit que la première valeur est 2, il envoie quelqu'un barrer tous les multiples non-triviaux de 2 (4, 6, 8 etc.). Il voit que la première valeur qui reste est 3, il envoie quelqu'un barrer tous les multiples

non-triviaux de 3 (6, 9, 12 etc.). Il voit que la première valeur qui reste est 5, il envoie un esclave barrer tous les multiples non-triviaux de 5, et ainsi de suite. Les entiers qui restent à la fin sont exactement les nombres premiers compris entre 2 et 100.

Écrivez un programme qui lit un entier n puis crée un tableau d'entiers de taille $n + 1$ dont il initialise toutes les cases à 1. Ensuite, pour chaque entier i allant de 2 à n,

- si la case d'indice i vaut 0, il n'y a rien à faire;
- si la case d'indice i vaut 1, alors on met à 0 les cases d'indice $2i$, $3i$, etc. sans faire aucune multiplication.

Votre programme affichera ensuite le nombre de cases d'indices 2 à n qui valent 1. Vérifiez que le crible d'Eratosthène produit les mêmes résultats que l'algorithme naïf. Comparez le temps d'exécution de vos deux programmes. Que pensez-vous de la quantité de mémoire nécessaire ?

4 Schéma de Horner

Une fonction polynomiale est une fonction qui a la forme

$$f(x) = \sum_{i=0}^{n-1} a_i x^i$$

où les coefficients a_i sont des valeurs arbitraires. Par exemple, la fonction $g(x) = 14x^2 + 13x + 32$ est une fonction polynomiale où $a_0 = 32$, $a_1 = 13$ et $a_2 = 14$.

Dans cet exercice, nous représenterons une fonction polynomiale par le tableau de ses coefficients. Par exemple, la fonction g ci-dessus sera représentée par le tableau $\{32, 13, 14\}$. (Remarquez que la taille du tableau est égale au degré du polynôme plus un.)

1. Écrivez une fonction `double eval(double x, double a[], int n)` qui calcule la valeur de la fonction polynomiale représentée par `a` au point `x` ; vous n'êtes pas autorisés à utiliser la fonc- tion `pow` de la bibliothèque standard. Combien votre fonction fait-elle de multiplications ? (Pensez à réutiliser la valeur de x^k pour calculer x^{k+1})

2. Remarquez que

   $$a_0+a_1x+a_2x^2+a_3x^3+\ldots+a_{n-1}x^{n-1} = a_0+x(a_1+x(a_2+x(a_3+x(\ldots(a_{n-2}+xa_{n-1})\ldots))))$$

   . Déduisez-en un algorithme qui évalue un polynôme en faisant $n - 1$ multiplications et implémentez-le. Vérifiez que votre programme calcule les mêmes valeurs que le programme écrit à la question précédente.