Partiel Algorithmique et Programmation 2020-2021

Exercice 1

Considérons une fonction déjà programmée dont le prototype est donné au-dessous
float f(int t);

- Expliquer le prototype de cette fonction. (1)

- Donner l'algorithme (en pseudo-code) d'une fonction qui trouve le maximum de f(t) dans l'intervalle [0,5]. Donner le code en C/C++ de cette fonction. (4)

- Donner l'algorithme d'une fonction qui donne la « dérivé » de la fonction f(t+1)-f(t) dans l'intervalle [0,5]. Donner le code en C/C++ de cette fonction.(4)

Solution

La fonction prend un paramètre entrée du type entier et elle sort une valeur en réel.

```
#include <iostream>
using namespace std;
float f(int t);
float df(int t);
float dfmax(int a,int b);
float fmax(int a,int b);
int main(int argc, const char * argv[]) {
    cout<<fmax(0,5)<<endl;
    cout<<dfmax(0,5)<<endl;

    return 0;
}

float f(int t){
    return t*t*2.1-t*0.3;
}
float fmax(int a,int b){
    int t=a;
    float valmax;
    while(t<=b){
        if(t==a){
            valmax=f(t);
        }
        else{
            if(valmax<f(t)){
                valmax=f(t);
            }
        }
        t++;
    }

    return valmax;
}
```

```
float df(int t){
    return f(t+1)-f(t);
}


float dfmax(int a, int b){
    int t=a;
    float valmax;
    while(t<=b){
        if(t==a){
            valmax=df(t);
        }
        else{
            if(valmax<df(t)){
                valmax=df(t);
            }
        }
        t++;
    }
    return valmax;
}
```

Exercice 2

Les résultats d'un examen est donnés en « A,B,C,D,E,F ». « A, B, C, D » signifient que l'étudiant a réussi cet examen. « E» signifie que l'étudiant a raté cet examen mais aura une chance de rattrapage. « F » signifie que l'étudiant a raté cet examen sans la chance de rattrapage.

Dans le code ci-dessous,
N représente le nombre d'étudiants.
Les tableaux resultat et num_etudiant stockent respectivement le résultat de chaque étudiant et son numéro identifiant.

Le tableau num_reussi devra stocker les identifiants des étudiants qui ont réussi l'examen.

```
#include <iostream>
using namespace std;

int compter(char x,char* res,int N);
float taux_reussie(char* res,int N);
void ecrire_num_reussi(int * tab,int*num ,char* res,int N);


int main(int argc, const char * argv[]) {
   char resultat[8]={'A','B','C','F','E','C','C','D'};
   int num_etudiant[]={12,22,128,54,58,465,47,78};
   int N=8;
   int i;

   char x='A';
   cout<<x<<":"<<compter(x,resultat,N)<<endl;
   cout<<"Taux de réussite: "<<taux_reussie(resultat,N)<<endl;

   int num_reussi[(int)(taux_reussie(resultat, N)*N)];
```

```
    ecrire_num_reussi(num_reussi,num_etudiant,resultat,N);
    for(i=0;i<(taux_reussie(resultat, N)*N);i++){
        cout<<num_reussi[i]<<endl;
    }

    return 0;
}
```

- Donner l'algorithme (en pseudo-code) d'une fonction qui donne le nombre d'étudiants ayant un certain résultat parmi « A,B,C,D,E,F ». Donner le code en C/C++ de cette fonction **int** compter(**char** x,**char*** res,**int** N);. (3)
- Donner l'algorithme d'une fonction qui donne le taux de réussite de cet examen. Donner le code en C/C++ de cette fonction **float** taux_reussie(**char*** res,**int** N);.(4)
- Donner l'algorithme (en pseudo-code) d'une fonction qui enregistre les identifiants des étudiants ayant réussi l'examen dans un tableau donnée par référence. Donner le code en C/C++ de cette fonction **void** ecrire_num_reussi(**int** * tab,**int***num ,**char*** res,**int** N); (4)

Solution

```
#include <iostream>
using namespace std;

int compter(char x,char* res,int N);
float taux_reussie(char* res,int N);
void ecrire_num_reussi(int * tab,int*num ,char* res,int N);


int main(int argc, const char * argv[]) {
    char resultat[8]={'A','B','C','F','E','C','C','D'};
    int num_etudiant[]={12,22,128,54,58,465,47,78};
    int N=8;
    int i;

    char x='A';
    cout<<x<<":"<<compter(x,resultat,N)<<endl;
    cout<<"Taux de réussite: "<<taux_reussie(resultat,N)<<endl;

    int num_reussi[(int)(taux_reussie(resultat, N)*N)];
    ecrire_num_reussi(num_reussi,num_etudiant,resultat,N);
    for(i=0;i<(taux_reussie(resultat, N)*N);i++){
        cout<<num_reussi[i]<<endl;
    }


    return 0;
}
int compter(char x, char* res,int N){
    int nb_x=0;int i;
    for(i=0;i<N;i++){
```

```
        if(res[i]==x){
            nb_x++;
        }
    }
    return nb_x;
}

float taux_reussie(char* res,int N){
    float ret=0;
    int i; char re[4]={'A','B','C','D'};
    for(i=0;i<4;i++){
        ret=ret+compter(re[i],res,N);
    }
    return ret/N;
}

void ecrire_num_reussi(int * tab,int*num ,char* res,int N){
    int i,j=0;
    for(i=0;i<N;i++){
        switch (res[i]) {
            case 'A':
            case 'B':
            case 'C':
            case 'D':
                tab[j]=num[i];
                j++;
                break;
            default:
                break;
        }
    }
}
```