11.5 Manipulations de caractères

SOLUTION DE L'EXERCICE 85:

Cf. solution de l'exercice 87.

SOLUTION DE L'EXERCICE 86:

```
#include<iostream>

using namespace std;

const int lg_string = 10;
typedef char vecteur[lg_string];

void remplace_car(vecteur string, char c, int k);
int length(vecteur string);

int main()
{
    vecteur string = "exemples";
    int n;
    char c;

    cin >> n >> c;
    remplace_car(string, c, n);
    cout << string << endl;
}

void remplace_car(vecteur string, char c, int k)
{

  if (k <= length(string)) {
        string[k - 1] = c;
    }
}

int length(vecteur string)
{
    int cpt;
    for (cpt = 0; string[cpt] != '\0'; ++cpt);
    return cpt;
}
```

SOLUTION DE L'EXERCICE 87:

```
#include<iostream>

using namespace std;

const int lg_string = 80;
typedef char vecteur[lg_string];
```

```
void n_prem_car(vecteur string, vecteur string_res, int n);
void n_dern_car(vecteur string, vecteur string_res, int n);
void n_car_pos_p(vecteur string, vecteur string_res, int n, int p);
int length(vecteur string);

int main()
{
    vecteur string = "abcdefghij";
    vecteur string_res;
    int n, p;

    cin >> n >> p;

    cout << "les " << n << " premiers caracteres du string = ";
    n_prem_car(string, string_res, n) ;
    cout << string_res << endl;

    cout << "les " << n << " derniers caracteres du string = ";
    n_dern_car(string, string_res, n) ;
    cout << string_res << endl;

    cout << "les " << n << " caracteres du string a partir de la position ";
    n_car_pos_p(string, string_res, n, p) ;
    cout << p << " = " << string_res << endl;
}

void n_prem_car(vecteur string, vecteur string_res, int n)
{
    int i, taille = length(string);

    for (i = 0; i < n and i < taille; ++i)
        string_res[i] = string[i];
    string_res[i] = '\0';
}

void n_dern_car(vecteur string, vecteur string_res, int n)
{
    int longueur;

    longueur = length(string);
    if (longueur > n) {
        for (int i = longueur - n; i <= longueur; ++i)
            string_res[i + n - longueur] = string[i];
    } else {
        for (int k = 0; k <= longueur; ++k)
            string_res[k] = string[k];
    }
}

void n_car_pos_p(vecteur string, vecteur string_res, int n, int p)
{
    int longueur, i;

    longueur = length(string);
    if (p + n - 1 <= longueur)
```

```
        longueur = p + n;
    for (i = p; i < longueur; ++i)
        string_res[i - p] = string[i];
    string_res[i] = '\0';
}

int length(vecteur string)
{
    int i;
    for (i = 0; string[i] != '\0'; ++i);
    return i;
}
```

SOLUTION DE L'EXERCICE 88:

```
#include<iostream>

using namespace std;

const int lg_max_string = 100;
typedef char myString[lg_max_string];

bool inclus(myString, myString);
int length(myString);

int main()
{
    myString str1 = "mai", str2 = "maeearrraroi";
    if (inclus(str1, str2))
        cout << str1 << "  ' est inclu dans '  " << str2;
    else
        cout << str1 << "  ' n'est pas inclu dans  '  " << str2;
}

bool inclus(myString str1, myString str2)
{
    int longueur1, longueur2, i = 0, j = 0;

    longueur1 = length(str1);
    longueur2 = length(str2);
    while (longueur1 - i <= longueur2 - j and i < longueur1) {
        if (str1[i] == str2[j])
            ++i;
        ++j;
    }
    return (i == longueur1);
}

int length(myString str)
{
    int i = 0;

    for (i = 0; str[i] != '\0'; ++i);

    return (i);
```

```
}
```

SOLUTION DE L'EXERCICE 89:

```
#include<iostream>

using namespace std;

const int lg_max_string = 100;
typedef char myString[lg_max_string];

void MAT_premiers(myString);
bool minuscule(char);
char majuscule(char);

int main()
{
    myString str = "essa  essa esss";

    cout << str << endl;
    MAT_premiers(str);
    cout << str << endl;

}

void MAT_premiers(myString str)
{
    bool nouveau_mot = true;

    for (int i = 0; str[i] != '\0'; ++i) {
        if (str[i] != ' ') {
            if (nouveau_mot and minuscule(str[i])) {
                str[i] = majuscule(str[i]);
            }
            nouveau_mot = false;
        } else {
            nouveau_mot = true;
        }

    }
}

bool minuscule(char car)
{
    return (car >= 'a' and car <= 'z');
}

char majuscule(char car)
{
    return (car + 'A' - 'a');
}
```

SOLUTION DE L'EXERCICE 90:

```
#include<iostream>

using namespace std;

const int lg_string = 20;
typedef char vecteur[lg_string];

int length(vecteur string);
void supdoub(vecteur string);

int main()
{
    int l;
    vecteur string = "abcddefgggggghijk";
    l = length(string);

    cout << '[';
    for (int i = 0; i < l; i++)
        cout << string[i];
    cout << ']' << endl;

    supdoub(string);

    cout << '[';
    for (int i = 0; i < l - 1; i++)
        cout << string[i];
    cout << ']' << endl;

}

int length(vecteur string)
{
    int i;
    for (i = 0; string[i] != '\0'; i++);
    return i;
}

void supdoub(vecteur string)
{
    int n = length(string);
    int i = 0;

    for (; i < n - 1 and string[i] != string[i + 1]; i++);

    for (int j = i + 1; j < n; j++)
        if (string[j] != string[i]) {
            i++;
            string[i] = string[j];
        }

    string[i + 1] = '\0';
}

// output:
```

```
// [abcddefgggggghijkk]
// [abcdefghijk      ]
//
```

SOLUTION DE L'EXERCICE 91:

```
#include<iostream>

using namespace std;

const int MAXV = 100;

void fandr(char[], char[], char[], int &);

int length(char string[]){
  int i ;
  for (i = 0; string[i] != '\0'; ++i) ;
  return i ;
}



int main()
{
    int n = 15, drop;

    char R[] = "Je", C[] = "Nous", T[] = "Ah! Nous suis le rois.";

    cout << T << endl;
    fandr(T, C, R, drop);
    cout << T << endl;
    cout << drop << endl;
}

void fandr(char V[], char C[], char R[], int &DROP)
{
    int j, k, RLONG = length(R), CLONG = length(C), n = length(V);
    DROP = 0;
    for (int i = 0; i < n; i++) {
        for (j = 0; j < CLONG and V[i + j] == C[j]; j++);
        if (j == CLONG)         // on a trouve une occurence
        {
            if (CLONG <= RLONG) {
                if (n + RLONG - CLONG > MAXV) {
                    DROP += n + RLONG - CLONG - MAXV;
                    n = MAXV;
                } else
                    n += RLONG - CLONG;
                for (k = n; k >= i + CLONG; k--)
                    V[k] = V[k - (RLONG - CLONG)];
            } else {
                for (k = i + RLONG; k <= n; k++)
                    V[k] = V[k + CLONG - RLONG];
                n -= CLONG - RLONG;
            }
```

```
            for (k = 0; k < RLONG; k++)
                V[k + i] = R[k];
        }
    }
}
```

SOLUTION DE L'EXERCICE 92:

```
#include<iostream>

using namespace std;

double convert(char[]);

int main()
{
    char T[] = "1024";
    double d;

    d = convert(T);
    cout << d;
}

double convert(char S[])
{
    double res = 0.0;

    for (int i = 0; S[i] != 0; i++) {
        res = res * 10.0 + S[i] - '0';
    }
    return res;
}
```

SOLUTION DE L'EXERCICE 93:

Un première solution :

```
#include<iostream>

using namespace std;
#include<math.h>

const int MAX = 100;

typedef char myString[MAX];

int main()
{
    myString s;
    int a, i;

    cout << "Entrez un entier : ";
    cin >> a;
```

```
    if (a == 0) {
        s[0] = '0';
        s[1] = '\0';
    } else {
      i = (log((double)a) / log(10.0)) + 1;
        s[i] = '\0';
        while (a != 0) {
            s[i - 1] = (a % 10) + '0';
            --i;
            a /= 10;
        }
    }
    cout << s << endl;
}
```

Une seconde solution avec une symétrie préalable :

```
#include<iostream>

using namespace std;
#include<math.h>

const int MAX = 100;

typedef char myString[MAX];

void symetrie(myString, int);

int main()
{
    myString s;
    int a, i = 0;

    cout << "Entrez un entier : ";
    cin >> a;

    if (a == 0) {
        s[0] = '0';
        s[1] = '\0';
    } else {
        i = 0;
        while (a != 0) {
            s[i] = (a % 10) + '0';
            ++i;
            a /= 10;
        }
        s[i] = '\0';
        symetrie(s, i);
    }
    cout << s << endl;
}

void symetrie(myString s, int n)
{
    int save;
```

```
    for (int i = 0; i < n / 2; ++i) {
        save = s[i];
        s[i] = s[n - i - 1];
        s[n - i - 1] = save;
    }
}
```

SOLUTION DE L'EXERCICE 94:

```
#include<iostream>

using namespace std;
const int long_max = 100;
typedef char myString[long_max];

int max_diff(myString);

int main()
{
  myString str="ABCDEEF";

  cout << " String :  " << str << endl;
  cout << "max_diff  =  " << max_diff(str) << endl ;
}

int max_diff(myString str)
{
  int i = 0, n = 1, n_max = 1, j;

  while (str[i] != '\0') {
    for (j = i; str[j] != str[i + n]; ++j);
    if (j == (i + n)) {
      n = n + 1;
      if (n > n_max)
        n_max = n;
    } else {
      n = n - (j - i) - 1;
      i = j + 1;
    }
  }
  return (n_max);
}
```