COMPILATION TP 1
EMSI - 4ÈME IIR
2017/2018

Prof. M. D. RAHMANI

Rappel: traitement de caractères

Les macros d'analyse de caractères:

Ces macros sont incluses dans la bibliothèque **<ctype.h>**.

Elles acceptent comme argument un **char** ou un **int** et retournent un entier différent de 0 si l'argument est compris dans les limites indiquées ci-dessous:

macros	**condition**
`isalpha(c)`	A-Z, a-z
`isupper(c)`	A-Z
`islower(c)`	a-z
`isdigit(c)`	0-9
`isxdigit(c)`	0-9, A-F, a-f
`isspace(c)`	blanc
`isalnum(c)`	0-9, A-Z, a-z

Série 1: TP généralités

- Exercice 1: Ecrire un programme qui vérifie que les parenthèses d'une chaîne, saisie au clavier, sont bien équilibrées.
- Le programme doit afficher l'un des 3 messages:
  - Les parenthèses sont bien équilibrées
  - Il manque, 'n' parenthèses ouvrantes
  - il manque, 'n' parenthèses fermantes

Exercice 1(Solution) : TP1_**Exo1.cpp**


```
#include<stdio.h>
#include<conio.h>  // pour getch()
#include<string.h>

int main() {

     int i,    // indice pour parcourir la chaine
     j=0;      //entier qui s'incrémente s'il rencontre '('
               // et se decremente si ')'
     char c,// le caractère courant
          ch[30];     //la chaine lue au clavier

    printf("Donnez votre chaine a analyser:\t");
    gets(ch);
    i=0;
    while (i<strlen(ch)){//boucle qui teste la fin de la chaine
              c=ch[i];
              if (c=='(') ++j;
              else if (c==')') --j;
              i++;
    }

    if(!j) printf("les parenthses sont bien equilibrees");
    else if(j<0)
       printf("il manque %d  parenthese ouvrante",-j);
       else printf("il manque %d parenthese fermante",j);

    getch();
    return 0;
}
```

Série 1: TP généralités

- Exercice 2: A l'aide d'une déclaration de type union, écrire une fonction `CTOA`, qui lit un caractère et renvoie le code ascii correspondant.

Exercice 2 (Solution) : TP1_Exo2.cpp


```
#include<stdlib.h>
#include<iostream>
using namespace std;

union
     {
       char c;
       unsigned i;
     } p;
```

```
void CTOA() {
     p.i=0;
     cout<<"Donner un caractere: ";
     cin>>p.c;
     cout<<"\n\tSon code ASCII est: "<<p.i;
     cout<<"\n";
}

int main() {

    CTOA();

    system("pause");

     return 0;

}
```

Série 1: TP généralités

- Exercice 3:
- Ecrire un programme ayant pour effet de synthétiser des expressions arithmétiques composées du seul chiffre 4, des quatre opérateurs +, -, *, / et les parenthèses (,).
- Par exemple : on peut écrire : « 2 = (4+4)/4 » et « 5=4+(4/4) ».
- On teste ce programme pour les entiers de 1 à 20.

Exercice 3

Solution : exo3.cpp

```
#include<stdio.h>
#include<stdlib.h>

// Synthèse d'expressions avec le chiffre 4 les parenthèses et les 4
//opérateurs

// la fonction exp reçoit un entier et donne une expression résultat
void exp(unsigned i) {
    unsigned n1=i/4; // n1 la partie entière
    unsigned n2=i%4; // le reste de la division par 4
    // d'une manière récursive on refait la même chose à n1 jusqu'à
    //  trouver une partie entière égale à 0
    if(n1!=0)
        {
            printf("4");
            if (n1>1) {printf("*(",n1); exp(n1); printf(")");}
        }
            if (n2!=0 && n1!=0) printf("+(");
             if (n2==1) printf("4/4");
             if (n2==2) printf("(4+4)/4");
             if (n2==3) printf("4-(4/4)");
             if (n2!=0 && n1!=0) printf(")");
 if(n1==0) return;
}

int main() {
   for(unsigned i=1; i<=20; printf("\n%i= ",i), exp(i), i++);
   system("pause");
   return 0;
}
```

Série 1: TP généralités

- Exercice 4:
- Ecrire un programme déterminant le nombre de la lettre ‘e’ (minuscule) dans un texte fourni au clavier.

Série 1: TP généralités

- Exercice 5:
- Ecrire un programme qui supprime toutes les lettres ‘e’ (minuscules) d’un texte fourni au clavier.

Série 1: TP généralités

- Exercice 6:
- Ecrire un programme qui lit au clavier un mot d'au plus 30 caractères et l'affiche à l'envers.

Série 1: TP généralités

- **Exercice 7:**

- Ecrire un programme qui lit un verbe du premier groupe et en affiche la conjugaison au présent, sous la forme :
- je chante
- tu chantes
- il chante
- nous chantons
- vous chantez
- ils chantent

- Le programme devra vérifier que le mot fourni se termine par « **er** ».

Analyse lexicale : AFD

- **Exercice 8:**

- Ecrire un programme qui implante un automate à états fini déterministe, qui lit une suite de '**a**' et indique si un nombre est constitué d'un nombre paire ou impaire de '**a**'.

Analyse lexicale

- **Exercice 9:** /* Comment sera traitée l'expression: a/*p */

```
#include <stdio.h>
#include <conio.h>
int main() {
        int a, *p;
        a = 123;
        p = &a;
        printf("%d\n", a/*p);
        printf("La ligne precedente ne contient pas %d\n", 1
        /* mais seulement la valeur de a "123" */);
//      printf("%d\n", a/ *p);  cette forme donne 1
        getch();
        return 0;
}
```

Analyse lexicale : AFD

- **Exercice 10:**

- Ecrire un programme qui implante un automate qui valide un commentaire à la C++ et affiche le contenu du commentaire.

  exemple : // un commentaire \n

Analyse lexicale : AFD

- **Exercice 11:**

- Ecrire un programme qui implante un automate qui valide un commentaire à la C et affiche le contenu du commentaire.

  exemple : /* un commentaire */

Analyse lexicale : AFD

- **Exercice 12:**

- Ecrire un programme qui implante les automates précédents des commentaires C et C++ et affiche le contenu du commentaire.

  exemple : /* un commentaire */ et // un commentaire \n