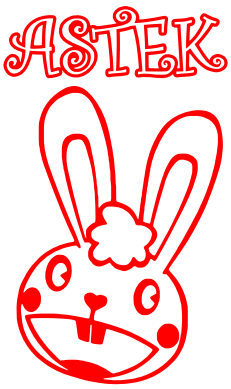

Programmation élémentaire

Bistromathique

Responsable de module wiart_mp@epitech.eu

Abstract: Ce document est le sujet du projet Bistromathique de Programmation élémentaire

Table des matières

Chapitre I

Consignes

- Le projet est à faire en binôme (2 et seulement 2).
- Vos exercices doivent être à la norme.
- Vous ne pouvez utiliser que les éléments vus à la piscine.
- Seul le rendu du chef de groupe sera ramassé.
- Le répertoire doit avoir un fichier auteur dans lequel vous devez mettre les deux logins.

```
1    (user@host h)cat auteur
2    login_1:login_2
3    (user@host h)
```

- Rendu :
  `Nom du répertoire de rendu : Bistromathique`

`Attention aux droits de vos fichiers et de vos répertoires`

Chapitre II

Sujet

- Il s'agit d'écrire un programme capable d'afficher le résultat de l'évaluation d'une expression arithmétique composée d'entiers de taille infinie exprimés dans une base quelconque. Ce programme traite les opérateurs suivant : "+-*/%", ainsi que les parenthèses. '(' ')'. Il gérera les priorités et les erreurs de syntaxe. L'ensemble des opérations est fait sur des entiers : 3/4*4=0.
- `usage : ./calc base opérateurs size_read`
- Exemples :

```
(user@host h)echo | ./calc
(user@host h)echo "3+6" | ./calc 0123456789 "()+-*/%" 3 ; echo
9
(user@host h)echo "3v6" | ./calc 0123456789 "{}vwxyz" 3 ; echo
9
(user@host h)echo "--++-6(12)" | ./calc 0123456789 "()+-*/%" 10 ; echo
syntax error
(user@host h)echo "--++-6*12" | ./calc 0123456789 "()+-*/%" 9 | cat -e ;
    echo
-72
(user@host h)echo "-(12-(4*32))" | ./calc 0123456789 "()+-*/%" 12 | cat -e
    ; echo
116
(user@host h)
(user@host h)echo "-(12-(4*32)" | ./calc 0123456789 "()+-*/%" 11 | cat -e ;
     echo
syntax error
(user@host h)
(user@host h)echo "-(&!-(;*!@))" | ./calc "~^@!;i &[]" "()+-*/%" 13 | cat -
    e ; echo
ii
(user@host h)echo "-(12*(13+15/5*(6/(12+14%(30%5+(10*25)-46)+16)-20)/43)
    *20)*(-(12-98*42)*(16+63-50/3))" | ./calc "0123456789" "()+-*/%" 84 |
    cat -e ; echo
-744629760
```

```
20    (user@host h)
```

Chapitre III

Détails techniques

- Un `main.c` et un `bistromathique.h` sont donnés dans la partie annexe. Il vous reste à coder la fonction evalexpr.
- En cas d'erreur de syntaxe, le programme affiche la chaîne de caractères définie par la macro SYNTAXE_ERROR_MSG.
- Vous ne pouvez utiliser que les fonctions : my_putchar, malloc, free.
- Vous pouvez poser vos questions dans le forum rubrique `B1-C-Prog Elem`
- Les programmes doivent être écrits en C (à la norme).
- Il devra y avoir un Makefile à la norme.
- L'exécutable doit s'appeler : 'calc' et se trouver dans le répertoire principal.

Chapitre IV

Tests Unitaires

Vous devrez également réaliser des tests unitaires, afin de tester votre projet. Pour cela vous réaliserez un script test que vous mettrez dans le dossier test. Votre script devra être executable peut importe le langage utilisé (pensez au shebang).

`gcov et lcov`

Chapitre V

Annexes

```
(user@host h)cat main.c
/*
** main.c for bistromathique
**
** Made by Charlie Root
** Login <rn@epita.fr>
**
** Started on Tue Oct 23 11:45:05 2001 Charlie Root
** Last update Mon Sep 17 12:00:27 2012 Mickael Wiart
*/

#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "bistromathique.h"

/*
** Remplacer cette ligne par un include de votre my.h
*/
void  my_putstr(char *);
int  my_strlen(char *);
int  my_atoi(char *);

static void check_base(char *base);
static void check_ops(char *ops);
static char *get_expr(unsigned size);

int  main(int ac, char **av)
{
  char  *expr;
  unsigned int size;

  if (ac != 4)
    {
      my_putstr("Usage : ");
      my_putstr(av[0]);
```

```
37      my_putstr(" base ops\"()+-*/%\" exp_len\n");
38      exit(1);
39    }
40  check_base(av[1]);
41  check_ops(av[2]);
42  size = my_atoi(av[3]));
43  expr = get_expr(size);
44  my_putstr(eval_expr(av[1], av[2], expr, size));
45  return (0);
46 }
```

```
1  static void check_base(char *b)
2  {
3   if (my_strlen(b) < 2)
4     {
5       my_putstr("Bad base\n");
6       exit(1);
7     }
8  }
9
10 static char *get_expr(unsigned int size)
11 {
12   char *expr;
13
14   if (size <= 0)
15     {
16       my_putstr("Bad expr len\n");
17       exit(2);
18     }
19   expr = malloc(size+1);
20   if (expr == 0)
21     {
22       my_putstr("could not alloc\n");
23       exit(3);
24     }
25   if (read(0, expr, size) != size)
26     {
27       my_putstr("could not read\n");
28       exit(4);
29     }
30   expr[size] = 0;
31   return (expr);
32 }
33
34 static void check_ops(char *ops)
35 {
36   if (my_strlen(ops) != 7)
37     {
38       my_putstr("Bad ops\n");
39       exit(5);
40     }
41 }
42
43 (user@host h)
44 (user@host h)cat bistromathique.h
45
46 /*
47 ** bistromathique.h for bistromathique in .
48 **
49 ** Made by Charlie Root
50 ** Login
```

```
51 **
52 ** Started on Tue Oct 23 11:48:35 2001 Charlie Root
53 ** Last update Tue Oct 23 11:52:38 2001 Charlie Root
54 */
55
56 /*
57 ** should be remove if you include stdlib.h (malloc.h does it)
58 */
59 void *malloc(unsigned int);
60
61 #define OP_OPEN_PARENT_IDX 0
62 #define OP_CLOSE_PARENT_IDX 1
63 #define OP_PLUS_IDX 2
64 #define OP_SUB_IDX 3
65 #define OP_NEG_IDX 3
66 #define OP_MULT_IDX 4
67 #define OP_DIV_IDX 5
68 #define OP_MOD_IDX 6
69
70 #define SYNTAXE_ERROR_MSG "syntax error"
71
72 char *eval_expr(char *base,char *ops,char *expr,unsigned int size);
```