Programmation Impérative 2 L2 Informatique

TD 1

Exercice I -

Ecrire un programme qui affiche la différence entre la moyenne arithmétique $A = \frac{a+b}{2}$ et la moyenne géométrique $G = \sqrt{a \times b}$ de deux nombre a et b dans les trois cas suivants:

1. les nombres sont fixes dans le programme.
2. Les nombres sont lus sur la ligne de commande.
3. Les nombres sont lus au clavier.

Exercice II - Voici un programme dans lequel manque un bout de code.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int x=0;
  int y=0;
  while(x<5){
      ###EMPLACEMENT DU BLOC###

      printf("%d%d ",x,y);
      x=x+1;
  }
  printf("\n");

  return EXIT_SUCCESS;
}
```

Essayez de faire correspondre le bloc de code candidat à gauche avec le résultat que vous verriez si le bloc était inséré.

Candidats:

1. `y=x-y;`
2. `y=y+x;`
3.
```
y=y+2;
if(y>4){
  y=y-1;
}
```
4.
```
x=x+1;
y=y+x;
```
5.
```
if(y<5){
  x=x+1;
  if(y<3){
    x=x-1;
  }
}
y=y+2;
```

Résultats possibles:

1. 22 46
2. 11 34 59
3. 02 14 26 38
4. 02 14 36 48
5. 00 11 21 32 42
6. 11 21 32 42 53
7. 00 11 23 36 410
8. 02 14 25 36 47

Exercice III - (Rendu de monnaie) Ecrire un programme qui prend en entrée une somme en euro (sans centime) et qui affiche le nombre minimal de billets et de pièces nécessaires pour la composer. On prendra à chaque fois le maximum de billets ou de pièces possible. Nous prenons en compte des billets de 500, 200, 100, 50, 20, 10 et 5 euros ainsi que des pièces de 2 et 1 euros.

Exemple:

$$1949 = 3 * 500 + 2 * 200 + 2 * 20 + 1 * 5 + 2 * 2$$

Exercice IV - Pas de questions stupides, reliez les questions aux réponses (en anglais). L'ordre des questions (et des réponses) peut avoir une importance.

1. Why are the characters numbered from 0? Why not 1?
2. Why ?
3. Why does it need a sentinel character? Doesn't it know how long the string is?
4. It doesn't know how long arrays are???
5. Does it matter if I use single quotes or double quotes?
6. So should I define my strings using quotes (") or as explicit arrays of characters?
7. Are there any differences between string literals and character arrays?
8. What does that mean?
9. What will happen if I try to modify a string literal?
10. A bus error? What the heck's a bus error?

a. No. Sometimes the compiler can work out the length of an array by analyzing the code, but usually C relies on you to keep track of your arrays.

b. Only one: string literals are constant. (but the character arrays must end by an additional `\0` to be considered as strings)

c. The computer will store the characters in consecutive bytes of memory. It can use the index to calculate the location of the character. If it knows that `c[0]` is at memory location 1,000,000, then it can quickly calculate that `c[96]` is at 1,000,000 + 96.

d. Usually you will define strings using quotes. They are called **string literals**, and they are easier to type.

e. Usually, it doesn't. C is not very good at keeping track of how long arrays are, and a string is just an array.

f. Yes. Single quotes are used for individual characters, but double quotes are always used for strings.

g. It depends on the compiler, but `gcc` will usually display a bus error.

h. C will store string literals in memory in a different way. A bus error just means that your program can't update that piece of memory.

i. The index is an offset: it's a measure of how far the character is from the first character.

j. It means that you can't change the individual characters once they are created.

Exercice V - Ecrire un programme qui lit une chaîne de caractères au clavier pouvant contenir au plus 2 caractères représentant les valeurs d'une carte à jouer. Les valeurs possibles des cartes sont *A, K, Q, J* ou les nombres de 1 à 10. Le programme devra afficher la valeur de la carte: 11 pour un as, 10 pour les figures, la valeur de la carte sinon.

Comment faire pour écrire un programme qui permette à l'utilisateur d'effectuer la même opération plusieurs fois pour connaître la valeur d'un ensemble de cartes ?

Exercice VI - Jouez au compilateur

Voici 4 portions de code, indiquez si chacune d'elle compile ou non. Si le code ne compile pas, indiquez pourquoi, s'il compile indiquez ce qu'il doit faire et si vous pensez que le code fait bien ce qu'il devrait.

Programme A

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

int main()
{
    int card = 1;
    if (card > 1)
        card = card - 1;
        if (card < 7)
            puts("Small card");
    else {
        puts("Ace!");
    }
    return EXIT_SUCCESS;
}
```

Programme B

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

int main()
{
    int card = 1;
    if (card > 1) {
        card = card - 1;
        if (card < 7)
            puts("Small card");
        else
           puts("Ace!");
     }
    return EXIT_SUCCESS;
}
```

Programme C

```
#include <stdio.h>
int main() {
    int card = 1;
    if (card > 1) {
        card = card - 1;
        if (card < 7)
            puts("Small card");
    }else
      puts("Ace!");

    return EXIT_SUCCESS;
}
```

Programme D

```
#include <stdio.h>
int main() {
    int card = 1;
    if (card > 1) {
        card = card - 1;
        if (card < 7)
            puts("Small card");
    else
       puts("Ace!");

    return EXIT_SUCCESS;
}
```

Exercice VII - Reliez chaque fonction avec sa description.

1. `strchr()`	a. Concaténer deux chaînes.
2. `strcmp()`	b. Trouver la position d'une chaîne dans une autre chaîne
3. `strstr()`	c. Touver la position d'un caractère dans une chaîne
4. `strcpy()`	d. Renvoyer la longueur de la chaîne
5. `strlen()`	e. Comparer deux chaînes
6. `strcat()`	f. Copier une chaîne dans une autre.

Exercice VIII - Voici une fonction avec des instructions manquantes, essayez de la compléter avec certains des éléments suivants: `strstr`, `tracks[i]`, `my`, `"Sinatra"`, `search_for`, `way`, `i`, `tracks[i]`.

```
void find_track(char search_for[])
{
   int i;
   for (i = 0; i < 5; i++) {
      if (........( ........ , ............. ))
        printf("Track %d: '%s'\n",........, ........... );

   }
}
```

Exercice IX - Choisissez la meilleure fonction `main` parmi les 4 suivantes:

A

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

int main()
{
   char search_for[80];
   printf("Search for: ");
   fgets(search_for, 80, stdin);
   find_track();

   return EXIT_SUCCESS;
}
```

B

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

int main()
{
    char search_for[80];
    printf("Search for: ");
    fgets(search_for, 79,stdin);
    find_track(search_for);

    return EXIT_SUCCESS;
}
```

C

```
#include <stdio.h>
int main() {
    char search_for[80];
    printf("Search for: ");
    fgets(search_for, 80, stdin);
    find_track(search_for);

    return EXIT_SUCCESS;
}
```

D

```
#include <stdio.h>
int main() {
    char search_for[80];
    printf("Search for: ");
    scanf(search_for, 80, stdin);
    find_track(search_for);

    return EXIT_SUCCESS;
}
```

Exercice X - Simplifiez l'écriture de ce programme (aide: pensez au `break`)

```
int main() {
    char card_name[3];
    puts("Enter the card_name: ");
    scanf("%2s", card_name);
    int val = 0;
    if (card_name[0] == 'K') {
        val = 10;
    } else if (card_name[0] =='Q') {
        val = 10;
    } else if (card_name[0] =='J') {
        val = 10;
    } else if (card_name[0] =='A') {
        val = 11;
    } else {
       val = atoi(card_name);
    }
    /* Check if the value is 3 to 6 */
    if ((val > 2) && (val < 7))
       puts("Count has gone up");
    /* Otherwise check if the card was 10, J, Q, or K */
    else if (val == 10)
       puts("Count has gone down");
    return EXIT_SUCCESS;
}
```

Exercice XI - Reliez les questions avec les bonnes réponses en anglais:

Questions:

1. Why would I use a switch statement instead of an if?
2. Does the switch statement have to check a variable? Can't it check a value?
3. What are the advantages of using a switch statement?
4. Can I check strings in a switch statement?
5. How can I be sure that the default option will not be selected one a case has been matched?

Réponses

a. No, you can't use a `switch` statement to check a string of characters or any kind of array. The `switch` statement will only check a single value.

b. If you are performing multiple checks on the same variable, you might want to use a `switch` statement.

c. Yes, it can. The `switch` statement will simply check that two values are equal.

d. You can put a `break` at the end of each case section to make the code easier to understand, even at the cost of some efficiency, but check that they are at the right places.

e. There are several. First: clarity. It is clear that an entire block of code is processing a single variable. That's not so obvious if you just have a sequence of `if` statements. Secondly, you can use fall-through logic to reuse sections of code for different cases.

Exercice XII - Est-ce que ce programme compile ? Que fait-il ?

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

int main() {
  char *cards = "JQK";
  char a_card = cards[2];

  cards[2] = cards[1];
  cards[1] = cards[0];
  cards[0] = cards[2];
  cards[2] = cards[1];
  cards[1] = a_card;
  puts(cards);

  return EXIT_SUCCESS;
}
```

Exercice XIII - Triangle Ecrire un programme qui affiche un triangle formé d'étoiles. La longueur du triangle - son nombre de lignes - sera indiqué en argument sur la ligne de commande.

```
    *
   ***
  *****
 *******
*********
```

Exercice XIV - Pas de questions stupides

1. Why didn't the compiler just tell me I couldn't change the string?
2. Why are string literals stored in read-only memory?
3. Do all operating systems enforce the read-only rule?
4. What does `const` actually mean? Does it make the string read-only?
5. Do the different memory segments always appear in the same order in memory?
6. I still don't understand why an array variable isn't stored in memory. If it exists, surely it lives somewhere?
7. If I set a new array to a string literal, will the program really copy the contents each time?
8. You keep saying "declaration". What does that mean?
9. Why is scanf() called scanf()?

a. When the program is compiled, all the references to array variables are replaced with the addresses of the array. So the truth is that the array variable won't exist in the final executable. That's OK because the array variable will never be needed to point anywhere else.

b. `scanf()` means *"scan formatted"* because it's used to scan formatted input.

c. It's down to the compiler. The final machine code will either copy the bytes of the string literal to the array, or else the program will simply set the values of each character every time it reaches the declaration.

d. Because they are designed to be constant. If you write a function to print "Hello World," you don't want some other part of the program modifying the "Hello World" string literal.

e. String literals are read-only anyway. The const modifier means that the compiler will complain if you try to modify an array with that particular variable.

f. Because we declared the cards as a simple `char*`, the compiler didn't know that the variable would always be pointing at a string literal.

g. They will always appear in the same order for a given operating system. But different operating systems can vary the order slightly. For example, Windows doesn't place the code in the lowest memory addresses.

h. The vast majority do. Some versions of `gcc` on Cygwin actually allow you to modify a string literal without complaining. But it is always wrong to do that.

i. A declaration is a piece of code that declares that something (a variable, a function) exists. A definition is a piece of code that says what something is. If you declare a variable and set it to a value (e.g., `int x=4;`), then the code is both a declaration and a definition.