I- Les différents types de données en MIPS:

1- Déclaration des valeurs (integers, floats, double, characters, and strings)
2- Lecture de différents types des valeurs
3- Affichage de différents types des valeurs

1- Déclaration de différents types des valeurs:

```
# defferent types declarations:
myInteger:      .word    12
myFloat:        .float   1.23
myDouble:       .double  12.3482321
myChar:         .byte    'c'
myString:       .asciiz  "myString data types declaration"
```

```
1  .data
2          # defferent types declarations:
3          myInteger:       .word  12
4          myFloat:         .float 1.23
5          myDouble:        .double 12.3482321
6          myChar:          .byte 'c'
7          myString:        .asciiz "myString data types declaration"
8
9  .text
10
```

2- Lecture de différents types des valeurs

a- INTEGERS :

```
#1- read integers
li $v0, 5
syscall
move $t0, $v0
```

```
.text
        #1- read integers
        li $v0, 5
        syscall
        move $t0, $v0
```

b- FLOATS

```
#1- read float numbers and save by default in $f0
  li $v0, 6
  syscall
```

```
 9  .text
10
11          #1- read integers and save by default in $f0
12          li $v0, 6
13          syscall
```

c- DOUUBLES

```
#2.1- read double from keyboard,
 # stored by default in register $f0
 li $v0, 7
 syscall
```

```
4   .text
5           #2.1- read double from keyboard,
6           # stored by default in register $f0
7           li $v0, 7
8           syscall
```

d- CHARACTERS

```
.data
        # defferent types declarations:
        inputChar:              .byte ' ' # declaring a one byte space
.text

        #2.2- read a character from the keyboard:
        li $v0, 12      # store the inputChar in $v0
        syscall

        # sotre and move character
        la $s0, inputChar    # generate address to store a byte
        sb $v0, inputChar   # store byte

        # display a character
        lb $a0, inputChar
        li $v0, 11      # display a character
        syscall
```

```
1  .data
2          # defferent types declarations:
3          inputChar:      .byte ' ' # declaring a one byte space
4  .text
5
6          #2.2- read a character from the keyboard:
7          li $v0, 12          # store the inputChar in $v0
8          syscall
9
10         # sotre and move character
11         la $s0, inputChar       # generate address to store a byte
12         sb $v0, inputChar       # store byte
13
14         # display a character
15         lb $a0, inputChar
16         li $v0, 11          # display a character
17         syscall
18
```

e- STRINGS

```
.data
        # defferent types declarations:
        myString:    .space 20 # bytes reservations (1 byte for one character)
.text

        # read string from keyboard
        la $a0, myString    # the string address
        li $a1, 20          # the sting size
        li $v0, 8           # reading string code
        syscall
```

```
1  .data
2          # defferent types declarations:
3          # bytes reservations (1 byte for one character)
4          myString:       .space 20
5  .text
6
7          # read string from keyboard
8          la $a0, myString          # the string address
9          li $a1, 20                # the sting size
10         li $v0, 8                 # reading string code
11         syscall
```

3- Affichage de différents types des valeurs

a- INTEGERS :

a.1. affichage d'un entier déclarer

```
.data
        # defferent types declarations:
        myInteger:   .word  12
.text
        #2.1- display declared integers numbers
        lw $a0, myInteger
        li $v0, 1
        syscall
```

```
1  .data
2          # defferent types declarations:
3          myInteger:       .word  12
4  .text
5          #2.1- display declared integers numbers
6          lw $a0, myInteger
7          li $v0, 1
8          syscall
```

a.2. affichage d'un entier stocker dans un registre

```
.data
        # defferent types declarations:
        myInteger:   .word  12
.text
        # a storage integer in a register
        lw $t1, myInteger

        #2.1- display declared integers numbers
        move $a0, $t1
        li $v0, 1
        syscall
```

```
1   .data
2           # defferent types declarations:
3           myInteger:      .word  12
4   .text
5           # a storage integer in a register
6           lw $t1, myInteger
7
8           #2.1- display declared integers numbers
9           move $a0, $t1
10          li $v0, 1
11          syscall
```

b- FLOATS

b.1. affichage d'un nombre réel déclarer

```
.data
        # defferent types declarations:
        myFloat:      .float 1.23
.text
        #2.1- display declared float numbers
        lwc1 $f12, myFloat
        li $v0, 2
        syscall
```

```
1   .data
2           # defferent types declarations:
3           myFloat:          .float 1.23
4   .text
5           #2.1- display declared float numbers
6           lwc1 $f12, myFloat
7           li $v0, 2
8           syscall
```

b.2. affichage d'un réel déclarer

```
.data
        # defferent types declarations:
        myFloat:      .float 1.23
.text
        #2.1- display storage float number in a register
        lwc1 $f1, myFloat
        add.s $f12, $f0, $f1
        syscall
```

```
.data
        # defferent types declarations:
        myFloat:        .float 1.23
.text
        #2.1- display storage float number in a register
        lwc1 $f1, myFloat
        add.s $f12, $f0, $f1
        syscall

```

c- DOUUBLES

c.1. affichage d'un nombre réel double précision déclarer:

```
.data
        # defferent types declarations:
        myDouble:   .double 1.23
.text
        #2.1- display declared double number:
        ldc1 $f0, myDouble
        add.d $f12, $f0, $f4
        li $v0, 3
        syscall
```

```
.data
        # defferent types declarations:
        myDouble:       .double 1.23
.text
        #2.1- display declared double number:
        ldc1 $f0, myDouble
        add.d $f12, $f0, $f4
        li $v0, 3
        syscall
```

c.2. affichage d'un nombre réel double précision stocker dans un registre:

```
.data
        # defferent types declarations:
        myFirstDouble:      .double 1.2
        mySecondDouble:     .double 12.12354
.text
        #2.1- display declared double number:
        ldc1 $f2, myFirstDouble
        ldc1 $f4, mySecondDouble
        add.d $f6, $f4, $f2
        # $f6 storage a double value
        # e.g. a result of an arithmetic operation
        add.d $f12, $f0, $f6
        li $v0, 3
        syscall
```

```
1   .data
2           # defferent types declarations:
3           myFirstDouble:   .double 1.2
4           mySecondDouble:  .double 12.12354
5   .text
6           #2.1- a simple addition of two doubles:
7           ldc1 $f2, myFirstDouble
8           ldc1 $f4, mySecondDouble
9           add.d $f6, $f4, $f2
10          # $f6 storage a double value
11          # e.g. a result of an arithmetic operation
12          add.d $f12, $f0, $f6
13          li $v0, 3
14          syscall
```

d- CHARACTERS

d.1- Affichage d'un caractère déclarer :

```
.data
        # defferent types declarations:
        myChar:    .byte  'c'
.text
        #2.1- display a declared character:
        la $a0, myChar
        li $v0, 4
        syscall
```

```
1   .data
2           # defferent types declarations:
3           myChar:     .byte   'c'
4   .text
5           #2.1- display a declared character:
6           la $a0, myChar
7           li $v0, 4
8           syscall
9
```

d.2- Affichage d'un caractère stocker dans un registre:

```
        .data
                # defferent types declarations:
                myChar:    .byte  'c'
        .text

                la $t1, myChar

                #2.2- display a character from a register:
                move $a0, $t1
                li $v0, 4
                syscall
```

```
1  .data
2          # defferent types declarations:
3          myChar:     .byte   'c'
4  .text
5
6          la $t1, myChar
7
8          #2.2- display a character from a register:
9          move $a0, $t1
10         li $v0, 4
11         syscall
12
```

e- STRINGS

e.1- Affichage d'une chaine de caractère déclarer :

```
.data
        # defferent types declarations:
        myString:    .asciiz "myString data type"
.text

        # display a string
        la $a0, myString
        li $v0, 4       # display a character
        syscall
```

```
1  .data
2          # defferent types declarations:
3          myString:        .asciiz "myString data type"
4  .text
5
6          # display a string
7          la $a0, myString
8          li $v0, 4        # display a character
9          syscall
```

e.2- Affichage d'une chaine de caractère stocker dans un registre:

```
.data
        # defferent types declarations:
        myString:    .asciiz "myString data type"
.text

        # display a string
        la $t2, myString
        move $a0, $t2
        li $v0, 4       # display a character
        syscall
```

```
1  .data
2          # defferent types declarations:
3          myString:        .asciiz "myString data type"
4  .text
5
6          # display a string
7          la $t2, myString
8          move $a0, $t2
9          li $v0, 4        # display a character
10         syscall
```

***** Bonne chance *****