



C1- Unix & C Lab Seminar

pool_c_d03

Day 03

First Steps



Instructions

Before You Go Further

- Turn in directory **pool_c_d03**.
- Respecting the norm takes time, but it is good for you. This way your code will respect the norm from the first written line to the last.
Read carefully the norm documentation. You should type “alt+i” instead of “tab”
- Do not care about the header. You don't have to add it into your file.
- You shall leave in your directory no other files than those explicitly specified by the exercises. If one of your files prevents the compilation with *.c, the robot will not be able to do the correction and you will have a 0.
- You might have to use the NULL keyword. This keyword is located in seddef.h
- **Do not turn-in a main() function unless it has been explicitly asked.**
- **You are only allowed to use the write function to do the following exercises.**



Remember it is always better to create your repository at the beginning of the day and to turn-in your work on a regular basis. Do not forget the permissions rights!



Exercise 1

Alpha (2pts)

Turn in: pool_c_d03/ex_01/alpha.c

Prototype: void alpha();

Write a function that displays the alphabet in uppercase on a single line, in ascending order from the letter 'A' followed by a '\n'

Example :

main.c :

```
void    alpha ();
int     main ()
{
    alpha ();
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex01
~/pool_c_d03> ./ex01
ABCDEFGHIJKLMNOPQRSTUVWXYZ
~/pool_c_d03>
```



man write;

man ascii;

What is a prototype? Why is it used here? =D

Oh ... And also ... All developers are **lazy**... No exception! =P



Exercise 2

Reverse Alpha (1pt)

Turn in: pool_c_d03/ex_02/revalpha.c

Prototype: void revalpha();

Write a function that displays the alphabet in uppercase on a single line, in descending order from the letter 'Z' followed by a '\n'

Example :

main.c :

```
void    revalpha () ;
int     main ()
{
    revalpha () ;
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex02
~/pool_c_d03> ./ex02
ZYXWVUTSRQPONMLKJIHGFEDCBA
~/pool_c_d03>
```



Exercise 3

True loop (2pts)

Turn in: pool_c_d03/ex_03/my_true_loop.c

Prototype: void my_true_loop(unsigned int n);

Write a function 'my_true_loop', taking a number as parameter. This function will print a number of '+' equal to this parameter followed by a '\n'

Example :

main.c :

```
void my_true_loop(unsigned int n);
int main()
{
    my_true_loop(5);
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex03
~/pool_c_d03> ./ex03
+++++
~/pool_c_d03>
```



Exercise 4

Conditions (2pts)

Turn in: pool_c_d03/ex_04/conditions.c

Prototype: void conditions(int n);

Write a function 'conditions' taking a number as parameter. This function will print '+' if the number is greater than 0, '-' if the number is lesser than 0 and 'O' otherwise.

Example :

main.c:

```
void    conditions(int n);
int     main()
{
    conditions(-564);
    conditions(564);
    conditions(0);
    return (0);
}
```

```

Terminal
~/pool_c_d03> cc *.c -o ex04
~/pool_c_d03> ./ex04
-+0~/pool_c_d03>
```



Exercise 5

my_aff_comb (3pts)

Turn in: pool_c_d03/ex_05/my_aff_comb.c

Prototype: int my_aff_comb();

Write a function that displays in the ascending order all the different combinations of three different digits in the ascending order.

Example :

main.c :

```
int my_aff_comb();
int main()
{
    my_aff_comb();
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex05
~/pool_c_d03> ./ex05
012, 013, 014, 015, 016, 017, 018, 019, 023, ..., 789~/pool_c_d03>
```

In this example, all results are not shown but, of course, you have to display all different combinations.

987 is not here because 789 is already there.

999 is not here because that number's digits are not all different from each other.



Exercise 6

my_putnbr (5pts)

Turn in: pool_c_d03/ex_06/my_putnbr.c

Prototype: void my_putnbr(int n);

Write a function 'my_putnbr' taking a number as parameter. This function will print this number on the standard output.

Example :

main.c :

```
void my_putnbr(int n);
int main()
{
    my_putnbr(42);
    my_putnbr(-42);
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex06
~/pool_c_d03> ./ex06
42-42~/pool_c_d03>
```

Exercise 7

my_aff_comb2 (2pts)

Turn in: pool_c_d03/ex_07/my_aff_comb2.c

Prototype: int my_aff_comb2();

Write a function that displays all the different combinations of two numbers between 0 and 99, in ascending order.

Example :

main.c :

```
int my_aff_comb2();
int main()
{
    my_aff_comb2();
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex07
~/pool_c_d03> ./ex07
00 01, 00 02, 00 03, 00 04, 00 05, ..., 01 99, 02 03, ..., 98 99~/pool_c_d03>
```




Exercise 8

my_aff_combn (3pts)

Turn in: pool_c_d03/ex_08/my_aff_combn.c

Prototype: int my_aff_combn(int n);

Write a function that displays all the different combinations of n digits in the ascending order like in the example.

Example for n = 2:

main.c:

```
int my_aff_combn(int n);
int main()
{
    my_aff_combn(2);
    return (0);
}
```

```
Terminal
~/pool_c_d03> cc *.c -o ex08
~/pool_c_d03> ./ex08
01, 02, 03, 04, 05, ..., 79, 89
~/pool_c_d03>
```