Version: 1.0

Selection

Square

Summary

Create a program to print rectangles of varying patterns based on user-defined dimensions.

#C

#Impaterive

#Unix

Contents

Chapter 1

Instructions

- The group WILL be registered to defense automatically.
- Do not cancel it, you won't get a second one.
- Any question concerning the subject would complicate the subject.
- You have to follow the submission procedures for all your exercises.
- This subject could change up to an hour before submission.
- The program must compile with the following flags: -Wall -Wextra -Werror; and uses `cc`.
- If your program doesn't compile, you'll get `0`.
- Your program must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a 0 if there is a norm error inside.
- You will have to handle errors coherently. Feel free to either print an error message, or simply return control to the user.
- Rushes exercises have to be carried out by group of 2, 3 or 4.
- The mandatory rush number for your team will follow this rule:
  Alphabetical Index of the first letter of a randomly selected team member's first name (from 1 to 26) modulo 5.
- You must therefore do the project with the imposed team and show up at Your defense slot, with all of your teammates.
- You project must be done by the time you get to defense. The purpose of defense is for you to present and explain your work.
- Each member of your group must be fully aware of the works of the project. Should you choose to split the workload, make sure you all understand what everybody's done. During the defense, you'll be asked questions and the final grade will be based on the worst explanations.
- It goes without saying, but gathering the group is your responsibility. You've got all the means to get in contact with your teammates: phone, email, carrier pigeon, spiritism, etc. So don't bother blurping up excuses. Life isn't fair, that's just the way it is.

- However, if you've really tried everything one of your teammates remains unreachable: do the project anyway, and we'll try and see what we can do about it during the defense. Even if the group leader is missing, you still have access to the submission directory.
- If you want bonus points, you may submit other subjects or be able to use program arguments to test your function.

Make sure the subject that was originally assigned to your group works perfectly: If a bonus subject works, but the original one fails the tests, you'll get 0.

Context

The C Piscine is intense. It's your first big challenge at 42 — a deep dive into problem-solving, autonomy, and community.

During this phase, your main objective is to build your foundation — through struggle, repetition, and especially **peer-learning** exchange.

In the AI era, shortcuts are easy to find. However, it's important to consider whether your AI usage is truly helping you grow — or simply getting in the way of developing real skills.

The Piscine is also a human experience — and for now, nothing can replace that. Not even AI.

For a more complete overview of our stance on AI — as a learning tool, as part of the ICT curriculum, and as a growing expectation in the job market — please refer to the dedicated FAQ available on the intranet.

Main message

- Build strong foundations without shortcuts.
- Really develop tech & power skills.
- Experience real peer-learning, start learning how to learn and solve new problems.
- The learning journey is more important than the result.
- Learn about the risks associated with AI, and develop effective control practices and countermeasures to avoid common pitfalls.

Learner rules:

- You should apply reasoning to your assigned tasks, especially before turning to AI.
- You should not ask for direct answers to the AI.
- You should learn about 42 global approach on AI.

Phase outcomes:

Within this foundational phase, you will get the following outcomes:

- Get proper tech and coding foundations.
- Know why and how AI can be dangerous during this phase.

Comments and example:

- Yes, we know AI exists — and yes, it can solve your projects. But you're here to learn, not to prove that AI has learned. Don't waste your time (or ours) just to demonstrate that AI can solve the given problem.
- Learning at 42 isn't about knowing the answer — it's about developing the ability to find one. AI gives you the answer directly, but that prevents you from building your own reasoning. And reasoning takes time, effort, and involves failure. The path to success is not supposed to be easy.
- Keep in mind that during exams, AI is not available — no internet, no smartphones, etc. You'll quickly realise if you've relied too heavily on AI in your learning process.
- Peer learning exposes you to different ideas and approaches, improving your interpersonal skills and your ability to think divergently. That's far more valuable than just chatting with a bot. So don't be shy — talk, ask questions, and learn together!
- Yes, AI will be part of the curriculum — both as a learning tool and as a topic in itself. You'll even have the chance to build your own AI software. In order to learn more about our crescendo approach you'll go through in the documentation available on the intranet.

✓ Good practice:

> I'm stuck on a new concept. I ask someone nearby how they approached it. We talk for 10 minutes — and suddenly it clicks. I get it.

✗ Bad practice:

> I secretly use AI, copy some code that looks right. During peer evaluation, I can't explain anything. I fail. During the exam — no AI — I'm stuck again. I fail.

Chapter 2

Foreword

Here are the lyrics of a famous TV show for everyone:

```
[Verse 1]
I wanna be the very best
Like no one ever was
To catch them is my real test
To train them is my cause

I will travel across the land
Searching far and wide
Each pokemon to understand
The power that's inside

[Chorus]
Pokemon! Gotta catch 'em all! It's you and me
I know it's my destiny,
Pokemon! Oh you're my best friend
In a world, we must defend
Pokemon! A heart so true
Our courage will pull us through,

You teach me and I'll teach you,
Pokemon! Gotta catch'em all

[Chorus]

Every challenge along the way
With courage I will face.
I will battle every day
To claim my rightful place.
Come with me,
The time is right,
There's no better team.
Arm in arm we'll win the fight!
It's always been our dream!

[Chorus]
```

I could bet you were singing right now, but it doesn't matter for the moment. And this subject is not related to Pokemon by the way...

Chapter 3

Main subject

	Exercise00	
RushOX		
Directory: *ex00/*		
Files to Submit: `main.c, ft_putchar.c, rush0X.c`		
Authorized: `write`		

- Files to submit: `main.c`, `ft_putchar.c` and your `rush0X.c`, '0X' represents the rush number. For example `rush00.c`.
- Those three files will be compiled together.
- Your file `ft_putchar.c` should include the function `ft_putchar`.

main.c

```
int main()
{
    rush(5, 5);
    return (0);
}
```

- You must therefore write the function `rush` taking two variables of type `int` as arguments, named respectively `x` and `y`. No need to say this function should be on the `rush0X.c` file.
- Your function `rush` should display (on-screen) a rectangle of `x` characters for width, and `y` characters for length.
- Your function should never crash or loop indefinitely.
- Your `main` will be modified during defense, to check if you've handled everything you're supposed to.

- Here's an example of test we'll perform:

Example of test with an other main.c

```c
int main()
{
    rush(123, 42);
    return (0);
}
```

Chapter 4

Rush 00

Terminal Output with rush(5, 3)

```
$>./a.out
o---o
|   |
o---o
$>
```

Terminal Output with rush(5, 1)

```
$>./a.out
o---o
$>
```

Terminal Output with rush(1, 1)

```
$>./a.out
o
$>
```

Terminal Output with rush(1, 5)

```
$>./a.out
o
|
|
|
o
$>
```

Terminal Output with rush(4, 4)

```
$>./a.out
o--o
|  |
|  |
o--o
$>
```

Chapter 5

Rush 01

Terminal Output with rush(5, 3)

```
$>./a.out
/***\
*   *
\***/
$>
```

Terminal Output with rush(5, 1)

```
$>./a.out
/***\
$>
```

Terminal Output with rush(1, 1)

```
$>./a.out
/
$>
```

Terminal Output with rush(1, 5)

```
$>./a.out
/
*
*
*
\
$>
```

Terminal Output with rush(4, 4)

```
$>./a.out
/**\
*  *
*  *
\**/
$>
```

Chapter 6

Rush 02

Terminal Output with rush(5, 3)

```
$>./a.out
ABBBA
B   B
CBBBC
$>
```

Terminal Output with rush(5, 1)

```
$>./a.out
ABBBA
$>
```

Terminal Output with rush(1, 1)

```
$>./a.out
A
$>
```

Terminal Output with rush(1, 5)

```
$>./a.out
A
B
B
B
C
$>
```

Terminal Output with rush(4, 4)

```
$>./a.out
ABBA
B  B
B  B
CBBC
$>
```

Chapter 7

Rush 03

Terminal Output with rush(5, 3)

```
$>./a.out
ABBBC
B   B
ABBBC
$>
```

Terminal Output with rush(5, 1)

```
$>./a.out
ABBBC
$>
```

Terminal Output with rush(1, 1)

```
$>./a.out
A
$>
```

Terminal Output with rush(1, 5)

```
$>./a.out
A
B
B
B
A
$>
```

Terminal Output with rush(4, 4)

```
$>./a.out
ABBC
B  B
B  B
ABBC
$>
```

Chapter 8

Rush 04

Terminal Output with rush(5, 3)

```
$>./a.out
ABBBC
B   B
CBBBA
$>
```

Terminal Output with rush(5, 1)

```
$>./a.out
ABBBC
$>
```

Terminal Output with rush(1, 1)

```
$>./a.out
A
$>
```

Terminal Output with rush(1, 5)

```
$>./a.out
A
B
B
B
C
$>
```

Terminal Output with rush(4, 4)

```
$>./a.out
ABBC
B  B
B  B
CBBA
$>
```

Chapter 9

Submission and peer-evaluation

Turn in your assignment in your Git repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.

As these assignments are not verified by a program, feel free to organize your files as you wish, as long as you turn in the mandatory files and comply with the requirements.

You need to return only the files requested by the subject of this project.