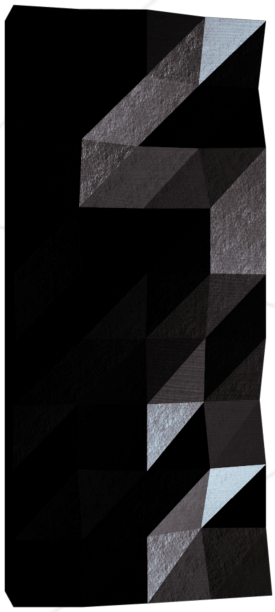

# C Piscine

## C 02

*Summary:* このドキュメントは、*42*の*C Piscine C 02*モジュール用の課題です。

*Version: 6*

# Contents

# Chapter I

# Instructions

- このページのみが正式な課題の指示です。噂に惑わされないようにしてください。
- この課題は提出前に変更される可能性がありますので、気をつけてください！
- ファイルとディレクトリへの適切な権限があることを確認してください。
- すべての課題で、**提出手順**に従ってください。
- 提出した課題の評価（レビュー）は、あなたの周りにいるPiscine受験者によって行われます。
- さらに、**Moulinette**（自動採点プログラム）にも評価されます。
- **Moulinette**は非常に厳格かつ厳密に評価を行います。完全に自動化されており、交渉の余地はありません。思わぬ減点を避けるためにも、細部まで丁寧に取り組んでください。
- **Moulinette**は融通が利きません。あなたのコードがNormに準拠していない場合、理解しようとしません。**Moulinette**は**norminette**というプログラムを使用して、ファイルがNormに準拠しているかチェックします。要約：**norminette**のチェックに通らない課題を提出しても意味がありません。
- 課題は難易度が低いものから高いものへと難易度順に並んでいます。前の課題が正しく解けていない場合、後の課題が解けていても評価されません。
- 禁止されている関数を使用することは不正行為とみなされます。不正行為者は**-42**という成績がつけられ、これは交渉の余地がありません。
- 私たちが具体的に**プログラム**を求める場合のみ、**main()**関数を提出する必要があります。
- **Moulinette**は以下のフラグでコンパイルします：**-Wall -Wextra -Werror**、**cc**を使用します。
- プログラムがコンパイルできない場合、**0**の評価になります。
- 課題で指定されたファイル以外は、ディレクトリに絶対に残さないでください。

- 質問がある場合は、まず右隣の仲間に聞いてみましょう。いなければ左隣の仲間に聞いてみてください。
- あなたの助けになる参考資料は**Google / man / インターネット / ...**です。
- Intranetのフォーラムの「C Piscine」セクションをチェックしてください。
- 課題の例（出力例）をよく読んでください。課題の説明文だけではすぐに分からない要件が隠れていることがあります。
- 頭を使ってください!!!

Norminetteは、 *-R CheckForbiddenSourceHeader* をオプションに追加しなければなりません。 Moulinetteも同様にこのオプションを使用します。

# Chapter II

# AI Instructions

## Context

C Piscineは強烈な体験です。42で迎える最初の大きな挑戦として、問題解決、自律性、そしてコミュニティの世界に深く飛び込むことになります。

このフェーズの主な目的は、試行錯誤や反復、そして特に**ピアラーニング**を通じた交流によって、自分の土台を築き上げることです。

AI時代において、近道を見つけることは簡単です。しかし、AIの利用が本当にあなたの成長を助けているのか、それとも真のスキルを身につける上で障害となっていないか、よく考えることが大切です。

Piscineはまたとない人との繋がりを経験する場でもあります。今のところ、それに取って代わるものは何もありません。AIでさえも。

学習ツールとして、ICTカリキュラムの一環として、そして労働市場での期待の高まりに応えるものとして、私たちのAIに対する考え方の詳細については、Intranet上の専用FAQをご覧ください。

## 主なメッセージ

- 近道をせず、強固な基礎を築く。
- 技術力と実践力を真に養う。
- 真のピアラーニングを体験し、学び方や新たな問題の解決方法を学び始める。
- 結果よりも学習の過程が重要である。
- AIに伴うリスクを学び、一般的な落とし穴を避けるための効果的な管理方法と対策を身につける。

## 学習者のルール:

- 課題には、AIに頼る前にまず自分の頭で考えること。
- AIに直接的な答えを求めないこと。
- AIに対する42のグローバルな方針を学ぶこと。

## このフェーズでの到達目標:

この基礎フェーズを終えることで、以下の点を達成できます。

- 技術とコーディングの適切な基礎を身につける。
- この段階においてAIが危険となりうる理由と、その危険性を理解する。

## コメントと具体例:

- 私たちはAIの存在を知っていますし、それがあなたのプロジェクトを解決できることも知っています。しかし、あなたは学ぶためにここにいるのであって、AIが学習したことを証明するためではありません。AIが特定の問題を解決できることを示すためだけに、あなた（や私たち）の時間を無駄にしないでください。
- 42での学習は、答えを知ることではありません。答えを見つけ出す能力を養うことです。AIは直接答えを与えてくれますが、それはあなた自身の論理的思考を妨げます。そして、論理的思考には時間と努力、そして失敗が伴います。成功への道は、決して楽なものではありません。
- Exam中はAIが利用できないことを心に留めておいてください。インターネットもスマートフォンもありません。学習過程でAIに過度に頼りすぎていた場合、すぐにそのことに気づくでしょう。
- ピアラーニングは、多様な考え方やアプローチに触れる機会を与え、対人スキルや多角的な思考能力を向上させます。これは、単にボットとチャットするよりもはるかに価値があります。ですから、恥ずかしがらずに、話しかけ、質問し、共に学びましょう！
- AIは学習ツールとして、またそれ自体がトピックとして、カリキュラムの一部になります。独自のAIソフトウェアを開発する機会さえあります。私たちの段階的なアプローチについてさらに学ぶためには、Intranetで利用可能なドキュメントを参照してください。

✓ 良い実践例:

> 新しいコンセプトの学習でつまづいています。近くの人にどうアプローチしたか尋ねてみます。10分ほど話していると、突然ひらめき、理解できました。

✗ 悪い実践例:

> こっそりAIを使い、良さそうなコードをコピーします。ピアレビューの際には何も説明できず、不合格になりました。ExamではAIがなく、また行き詰まり、不合格になりました。

# Chapter III

# Foreword

Here is an excerpt from a discussion in the *Silicon Valley* series:

```
- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emacs.
- Oh, God, help us! Okay, uh you know what? I just don't think this is
  going to work. I'm so sorry. Uh, I mean like, what, we're going to bring
  kids into this world with that over their heads? That's not really fair
  to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no
  way I'm going to be with someone who uses spaces over tabs.
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! - (DOOR SLAMS) - (BANGING)

. . .

(RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time. I'm okay, though.
- See you around, Richard.
- Just making a point.
```

Hopefully, you are not required to use emacs and your space bar to complete the following exercises.

# 今日の基準値

この課題をクリアするための基準値は50%です。

どの問題まで解けばこの基準に到達できるか、また、さらに多くの問題に挑戦するかの判断はあなた次第です。

# Chapter IV

# Exercise 00 : ft_strcpy

|  | Exercise 00 |
|---|---|
| ft_strcpy | |
| 提出するディレクトリ：$ex00/$ | |
| 提出するファイル：`ft_strcpy.c` | |
| 使用可能な関数：`None` | |

- `strcpy`関数の動作を再現しなさい。(man strcpy)
- プロトタイプ例：

```
char    *ft_strcpy(char *dest, char *src);
```

# Chapter V

# Exercise 01 : ft_strncpy

|  | Exercise 01 |
|---|---|
| ft_strncpy | |
| 提出するディレクトリ：$ex01/$ | |
| 提出するファイル：`ft_strncpy.c` | |
| 使用可能な関数：`None` | |

- `strncpy`関数の動作を再現しなさい。(man strncpy)
- プロトタイプ例：

```
char    *ft_strncpy(char *dest, char *src, unsigned int n);
```

# Chapter VI

# Exercise 02 : ft_str_is_alpha

|  | Exercise 02 |
|---|---|
| ft_str_is_alpha | |
| 提出するディレクトリ：$ex02/$ | |
| 提出するファイル：`ft_str_is_alpha.c` | |
| 使用可能な関数：`None` | |

- パラメータとして与えられた文字列に、英字のみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成してください。
- プロトタイプ例：

```
int ft_str_is_alpha(char *str);
```

- `str`が空の場合、関数が1を返すように実装してください。

# Chapter VII

# Exercise 03 : ft_str_is_numeric

|  | Exercise 03 |
|---|---|
| ft_str_is_numeric | |
| 提出するディレクトリ：$ex03/$ | |
| 提出するファイル：`ft_str_is_numeric.c` | |
| 使用可能な関数：`None` | |

- パラメータとして与えられた文字列に、数字のみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成してください。
- プロトタイプ例：

```
int ft_str_is_numeric(char *str);
```

- `str`が空の場合、関数が1を返すように実装してください。

# Chapter VIII

# Exercise 04 : ft_str_is_lowercase

|  | Exercise 04 |
|---|---|
| ft_str_is_lowercase | |
| 提出するディレクトリ：$ex04/$ | |
| 提出するファイル：`ft_str_is_lowercase.c` | |
| 使用可能な関数：`None` | |

- パラメータとして与えられた文字列に、小文字のアルファベットのみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成してください。
- プロトタイプ例：

```
int ft_str_is_lowercase(char *str);
```

- `str`が空の場合、関数が1を返すように実装してください。

# Chapter IX

# Exercise 05 : ft_str_is_uppercase

|  | Exercise 05 |
|---|---|
| ft_str_is_uppercase | |
| 提出するディレクトリ：$ex05/$ | |
| 提出するファイル：`ft_str_is_uppercase.c` | |
| 使用可能な関数：`None` | |

- パラメータとして与えられた文字列に、大文字のアルファベットのみが含まれる場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成してください。
- プロトタイプ例：

```
int ft_str_is_uppercase(char *str);
```

- `str`が空の場合、関数が1を返すように実装してください。

# Chapter X

# Exercise 06 : ft_str_is_printable

|  | Exercise 06 |
|---|---|
| ft_str_is_printable | |
| 提出するディレクトリ：$ex06/$ | |
| 提出するファイル：`ft_str_is_printable.c` | |
| 使用可能な関数：`None` | |

- パラメータとして与えられた文字列に、表示文字のみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成してください。
- プロトタイプ例：

```
int ft_str_is_printable(char *str);
```

- `str`が空の場合、関数が1を返すように実装してください。

# Chapter XI

# Exercise 07 : ft_strupcase

|  | Exercise 07 |
|---|---|
| ft_strupcase | |
| 提出するディレクトリ：$ex07/$ | |
| 提出するファイル：`ft_strupcase.c` | |
| 使用可能な関数：`None` | |

- 文字列にあるすべての文字を、大文字に変換する関数を作成してください。
- プロトタイプ例：

```
char    *ft_strupcase(char *str);
```

- 関数が`str`を返すように実装してください。

# Chapter XII

# Exercise 08 : ft_strlowcase

|  | Exercise 08 |
|---|---|
| ft_strlowcase | |
| 提出するディレクトリ：$ex08/$ | |
| 提出するファイル：`ft_strlowcase.c` | |
| 使用可能な関数：`None` | |

- 文字列にあるすべての文字を、小文字に変換する関数を作成してください。
- プロトタイプ例：

```
char    *ft_strlowcase(char *str);
```

- 関数が`str`を返すように実装してください。

# Chapter XIII

# Exercise 09 : ft_strcapitalize

|  | Exercise 09 |
|---|---|
| ft_strcapitalize | |
| 提出するディレクトリ：$ex09/$ | |
| 提出するファイル：`ft_strcapitalize.c` | |
| 使用可能な関数：`None` | |

- 各単語の最初の文字を大文字に変換し、それ以外の文字を小文字に変換する関数を作成してください。
- 単語とは英数字の文字列です。
- プロトタイプ例：

```
char    *ft_strcapitalize(char *str);
```

- 関数が、`str`を返すように実装してください。
- 例：

```
hi, how are you? 42words forty-two; fifty+and+one
```

- 上記の入力に対する出力は、以下のようになります。

```
Hi, How Are You? 42words Forty-Two; Fifty+And+One
```

# Chapter XIV

# Exercise 10 : ft_strlcpy

|  | Exercise 10 |
|---|---|
| ft_strlcpy | |
| 提出するディレクトリ：$ex10/$ | |
| 提出するファイル：`ft_strlcpy.c` | |
| 使用可能な関数：`None` | |

- `strlcpy`関数の動作を再現しなさい。(man strlcpy)
- プロトタイプ例：

```
unsigned int ft_strlcpy(char *dest, char *src, unsigned int size);
```

# Chapter XV

# Exercise 11 : ft_putstr_non_printable

|  | Exercise 11 |
|---|---|
| ft_putstr_non_printable | |
| 提出するディレクトリ：$ex11/$ | |
| 提出するファイル：`ft_putstr_non_printable.c` | |
| 使用可能な関数：`write` | |

- 与えられた文字列を標準出力に出力する関数を作成してください。文字列に出力不可能な文字が含まれている場合は、16進数（小文字）にして、その文字の前にバックスラッシュ“\”をつけて出力してください。
- 例：

```
Hello\nHow are you?
```

- 関数の出力例：

```
Hello\0aHow are you?
```

- プロトタイプ例：

```
void    ft_putstr_non_printable(char *str);
```

# Chapter XVI

# Exercise 12 : ft_print_memory

|  | Exercise 12 |
|---|---|
| ft_print_memory | |
| 提出するディレクトリ：$ex12/$ | |
| 提出するファイル：`ft_print_memory.c` | |
| 使用可能な関数：`write` | |

- メモリ領域を標準出力に出力する関数を作成してください。
- メモリ領域は、以下のように、3つの列に分けて出力してください。
  - 行の最初の文字のアドレス（16進数で表したアドレス）と、':' が出力されるようにしてください。
  - 以下の例を参照し、2文字ごとにスペースを含む16進数に、必要に応じて追加のスペースを入れてください。
  - アドレスにある要素を、表示文字に変換してください。
- 文字が表示文字ではない場合は、ドット“.”に置き換えてください。
- 各行は、16文字で構成されるようにしてください。
- **サイズ**が0の場合は何も出力しないでください。

- 例：

```
$> ./ft_print_memory
000000010a161f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin
000000010a161f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo
000000010a161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
000000010a161f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.
000000010a161f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..
000000010a161f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000           ..lol.lol. .
$> ./ft_print_memory | cat -te
0000000107ff9f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin$
0000000107ff9f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo$
0000000107ff9f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on $
0000000107ff9f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.$
0000000107ff9f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..$
0000000107ff9f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000           ..lol.lol. .$
$>
```

- プロトタイプ例：

```
void    *ft_print_memory(void *addr, unsigned int size);
```

- 関数が、addrを返すように実装してください。

# Chapter XVII

# Submission and peer-evaluation

課題は、いつも通りGitリポジトリに提出してください。リポジトリ内の提出物のみが、レビュー中の評価対象となります。ファイル名が正しいかどうか、必ず確認してください。

この課題の指示で、明示的に求められたファイルのみを提出してください。