第1章：達人の哲学

概要

あなたを達人プログラマーたらしめるのは、

- 問題を大きなコンテキストで考えるアプローチ
- その解決手段についての考え方・スタイル・哲学
- 責任をまっとうすること

実践的なプログラミングは達人的な思考哲学より生まれる

あなたの人生

業界には多くの機会が転がり落ちている。すべてはあなたのパッション。

> あなたには現状を打破する力がある

猫がソースコードを食べちゃった

誤りは起こるもの。誤りを認め、正直かつ単刀直入に対処すべし。
ある物事が達成できないとき、必要なもの・やるべきことを提案し、要求する。

> いい加減な言い訳よりも対策を用意すること

！チャレンジ！

ソフトウェアのエントロピー

宇宙と同様に、ソフトウェアも時間とともに無秩序になっていく性質を持つ。
悪い設計・誤った意思決定・質の低いコードは発見と同時に修復しなければならない。
修復できなければ、コメント・GitHubのIssueなど目印を残すことで被害の拡大を少しでも抑える行動を起こす。

1箇所の綻びがあれば、それ以降のコードもそれに追随して腐敗していく。汚染の連鎖を作らない。

> 割れた窓を放置しておかないこと

！チャレンジ！

石のスープとゆでガエル

道理にかなった小さな要求から実現した小成果を見せ、徐々に要求を大きくしていく。
うまくいき、未来が少し見えているものには、周囲は飛びついてくる。

ただし、常に大きな視点でものごとを見なければ、逆に騙されてしまう。

> 変化の触媒たれ
> 大きな構想を忘れないようにすること

！チャレンジ！

十分によいソフトウェア

バグのないソフトウェアを製造することはできない。
要求仕様・パフォーマンス・プライバシー・セキュリティなどの基本的要求が「十分に良い」かどうかの判定にユーザーを巻き込むこと。

多くのユーザーは目先の価値を求める。早期に成果物を提供することで、解決策を導くフィードバックを受けられる。

> 品質要求を明確にすること

！チャレンジ！

あなたの知識ポートフォリオ

IT分野ではあなたの持つ知識や経験は新しい技術や言語の登場により、かんたんに陳腐化する。
その中では、学習するという能力が重要な戦略的資産。

株式のポートフォリオのように、自らの技術ポートフォリオを作り上げる。

- 現在作業で使う特定技術のすべての詳細を知る。
- 関係ない分野のスキルについても手を広げる。
- リスク分散のため、特化しすぎない。

具体的には、

- 毎年1言語以上を学習

- 月1冊の技術書読書
- 技術書以外の書籍により、対人力を磨く
- ユーザーグループに参加
- GUI→CUIなど、異なる環境に触れる

これらを貪欲に、批判的に吟味して吸収し続ける

- なぜなぜ
- 利益の流れを追う
- コンテキストはなにか

> あなたの知識ポートフォリオに対して定期的な投資を行うこと
> 見聞きしたものごとを批判的な目で分析すること

！チャレンジ！

伝達しよう！

いかなる素晴らしいアイディアも伝えることができなければ何の意味もない。
言語もひとつのプログラミング言語として考え、DRY原則やETC・自動化などの設計原則に則ることで、伝達を円滑にする。

有効となる例

- 聞き手のニーズや興味を知る
- 言いたいことを先に練り上げる
- 適切なタイミングで伝える
- 伝えるスタイルを相手に合わせる
- 見栄えを良くする
- 聞き手を巻き込む
- 聞き手になる
- 相手の立場になる
- ドキュメントを残す

> 日本語をもうひとつのプログラミング言語として考えること
> 伝えることがらと、伝える方法は車の両輪だと考えること
> ドキュメントは付け足すものではなく、作り上げるものである

！チャレンジ！