

# 最速の仕事術は プログラマーが 知っている

**Programmer's  
SPEED HACKS**

**清水亮**  
SHIMIZU RYO



CrossMedia  
Publishing



## はじめに

---

### 「最速」

それはプログラマーなら誰もが心がけているキーワードである。

プログラマーにとって最も重要な関心事は、効率化だ。

熟練したプログラマーほど実際にはキーボードに向かわなくなるという。

なぜか？

それは、プログラミングに熟達すればするほど、コードをガリガリ書く時間よりも、むしろ書かない時間にこそ仕事の真髄があるとわかるからだ。

熟練プログラマーの仕事は一瞬で終わる。素人が数か月、ああでもないこうでもないで大騒ぎしてやっと終わるような仕事も、ものの数時間で終わらせるスーパープログラマーは確かに存在する。

世界でもプログラマーほど個人の生産性の違いに大きな差がある職業も珍しい。

執筆時点で、世界最速の男は100メートルを9秒58で走るウサイン・ボルトだ。彼が人類最速の男だとしよう。筆者は趣味でジョギングをしているが、1キロ走るのにだいたい5分かかる。ということはそのペースだと100メートルに30秒はかかることになる。スプリントレースとジョギングでは大きく異なるが、同じ「走る」という行為で比較すると、人類最速の男

と、一般的中年男性の運動能力には、ざっと 10 倍の差があるということがわかる。

だが熟練したプログラマーは、軽々と素人の何百倍、何億倍という効率の差を見せる。

例えば、こんなことがあった。

アメリカのボストンの北にある大学で特別講義をやったときのことだ。

筆者は生徒たちを前に、「今から 90 秒以内にゲームを作って見せます」と言って彼らに時間を測るよう言った。

それから実際にその場でゲームをプログラミングした。

実際に「完成」と言うまでに要した時間は 46 秒だった。

すると生徒の一人がたまらず笑い始めた。

その笑いはなぜか向こうの教師にも広がり、しまいには教室全体が笑いに包まれた。

「どういうことだ？」

筆者がややムツとして聞くと、最初に笑い始めた彼が言った。

「いや、実は我々はこの 2 週間、ゲームプログラミングの課題を与えられて、これまで散々四苦八苦しなうらややくできたというのに、あなたは 46 秒でやってしまったから驚いて笑ってしまったんです」

これはとても特殊な例だろうか。

筆者はそうは思わない。

プログラミングの世界では、熟練したプログラマーのたった一瞬の思いつきが世界を永遠に変えてしまうことがある。

Windows、Mac、iPhone、Facebook、Twitter……そうした数々のものを生み出したのはみんなプログラマーである。

プログラマーは効率を最優先する。

高速に仕事を終えることが、プログラマーにとって最大の報酬だ。

そしてプログラマーは自らが考えた「手抜き」の方法を、喜んでほかのプログラマーと共有する。

それを繰り返すことで、我々プログラマーは進化してきた。

この進化のスピードは、極めて速い。

19世紀末にアメリカのハーマン・ホラリスという発明家が初めて自動集計器を作ったとき、人力で13年かかってしまう集計を、わずか18か月で終えたという。つまり人間に対して8.6倍の効率で集計ができた。現在、100ます計算で成人男性とコンピュータが速度を競うとすると、成人男性が平均2分であるのに対し、コンピュータは25ナノ秒で計算を終える。つまり48億倍の速度ということになる。この100年で、コンピュータの処理速度は5.6億倍にまで高速化した。普通の人間が直感で理解できるスピードを超えた進歩を遂げているのがコンピュータの世界なのだ。

人間が創り出した最も高速な移動手段は、時速2万8000キロメートルで飛び、月まで到達したサターンV型ロケットだが、これは人間の歩行が時速4kmであることを考えると高々7000倍でしかない。これに対し、コンピュータの計算能力は人間の計算能力の実に48億倍に達するわけだ。

そのコンピュータを自在に操るプログラマーにとって、進化

についていけないことはすなわち死を意味する。そしてコンピュータの進化とはすなわち、効率化・高速化である。

プログラマーだけが複雑な情報処理を最速に終わらせる方法を知っているのである。

そして現代、仕事といえば大半が情報処理だ。

ミーティング、プレゼンテーション、リサーチ、それらはすべて情報処理の異なる側面にすぎない。

ホワイトカラーの仕事のなかで、情報処理でないものを探すほうが難しいほどだ。

であれば、最速の仕事術をプログラマーが知っているというのは言いすぎにはならないだろう。

あるとき、筆者は電通の役員室でこんなことを言ったことがある。

「プログラミングができないというのは、頭の使い方を知らないのに等しい」

筆者は仕事柄、MBA を持つビジネスエリートや、弁護士、コンサルタントとも日常的に接している。しかし MBA を持っていようと、弁護士資格を持っていようと、プログラマーよりも頭の使い方を心得ているとは必ずしも言えまい。

弁護士はヘマをして裁判に負けるし、MBA ホルダーのコンサルタントは提案した事業が失敗しても責任はとらない。

ところがプログラマーは、プログラマーだけは、自分の書いたプログラムにバグ（誤り）があるとき、それは100%自分の責任なのである。そしてバグのないプログラムは決して存在しないのだ。

すなわち、プログラマーはどれだけ思い上がろうとも、プログラムを書いている限り、自分の知性の限界を思い知らされ続けることになる。それもほかならない、自らの唯一の相棒であるコンピュータによってだ。

したがってプログラマーは常に自分の知性の限界を超え続けなければならないことを宿命づけられているのである。

筆者とて、自分が完璧な存在だとは露ほどにも思ったことはない。

ただ、情報処理の方法についてほかの職業に負けるのは恥だと考えている程度にはプライドがある。

そして前述したように、ホワイトカラーの大半の仕事は情報処理である。

したがって、最速の仕事術を常に求めているのはプログラマーであるということになる。

本書は、そんな不遜なプログラマーによる仕事術の本だ。

プログラマーの思考法は普通の仕事をしては身につかない。しかし、普通の仕事をしている人と、共通する点も少なくない。

ぜひ、プログラマーでない人も、我々がどのように物事を考えているかを知り、日々の生活に活かしてもらいたい。

# 1

はじめに ..... 3

## 速くてムダのない シンプル仕事術

1 辞書登録でメールを 5 秒で送信 ..... 17  
*DRY (Don't Repeat Yourself)*

2 タイピングを見直して  
生産性 3 割アップ ..... 26  
*かな入力と親指ソフトキーボード*

3 主語・動詞・目的語で  
議事録は書け ..... 32  
*PIE (Program Intently and expressively)*

4 紙の資料でもリンクを張れ ..... 36  
*KISS (Keep It Simple, Stupid!)*

5 長文に Word を使うな ..... 40  
*テキストエディタと Markdown 記法*

6 企画を宇宙戦艦にするな ..... 46  
*YAGNI (You aren't gonna need it)*

7 プレゼン資料は直前に書け ..... 48  
*新製品発表会とライブコーディング*

COLUMN 1 エジソンは手抜きの天才だった ..... 51



# 頭がクリアになる 情報の整理法

- 1 情報を覚えておく時代はもう終わった ..... 55  
*ユビキタスキャプチャ*
  - 2 情報の整理は見出しのつけ方が 9 割 ..... 60  
*データ構造*
  - 3 目当てのファイルが  
瞬時に見つかるタイトルづけ ..... 64  
*ディレクトリとツリー構造*
  - 4 1 秒でスクショ、2 秒で共有 ..... 69  
*Gyazo (ギャゾー)*
  - 5 書類トレイの 2 つのモード ..... 72  
*FIFO (先入れ先出し) と LIFO (後入れ先出し)*
  - 6 フォルダに穴をあけて取り出さずに  
サインする ..... 75  
*最適化とループ処理*
  - 7 Web に本当に大事な情報は  
載っていない ..... 83  
*Web の発達と情報の格差*
  - 8 Google の本当の使い方 ..... 86  
*検索ツールと Internet Archive*
  - 9 一次情報を集める最も効率のいい方法 ..... 90  
*情報発信とインターネットワーク*
  - 10 クリエイティビティは  
移動距離に比例する ..... 97  
*ゲームクリエイターの教え*
- COLUMN 2 数学少年カールとアルゴリズム ..... 100

## 3

致命的なミスを防ぐ  
賢いダンドリ

- 1 危機を事前に察知する  
プログラマーの本能 ..... 105  
*デバッグ*
  - 2 外部チェックを入れて  
大失敗を回避する ..... 108  
*社外取締役とテストエンジニア*
  - 3 簡単なタスクをど忘れしないための  
リマインダー ..... 111  
*バグ追跡システム*
  - 4 そつのない幹事の思考法 ..... 113  
*アーキテクチャとダンドリ*
  - 5 匿名チャットで経営会議 ..... 118  
*Flat と Slack*
  - 6 問題解決は切り分けることから始まる ..... 122  
*原因特定プロセス*
  - 7 「順調です」は信じない ..... 126  
*産業スパイとプロジェクト管理*
  - 8 想定外を想定しておく ..... 129  
*フェイルセーフ*
  - 9 今日と明日に急な予定は入れない ..... 131  
*プリフェッチ*
  - 10 客からの電話には出ない ..... 133  
*デメテルの法則とシリアライズ*
- COLUMN 3 独身 OL とループの最適化 ..... 138

## 4

# チームの成果を 最大化する仕組み

- 1 好かれるリーダーより  
やりきるリーダーになれ ..... 143  
*「オフィサー」の意味*
  - 2 他人を思い通りに動かすスキル ..... 150  
*πρόγραμμα (プログラム)*
  - 3 リーダーは「働いたら負け」 ..... 152  
 *$\cos \theta \cdot F$*
  - 4 仕事は細分化してラインをつくれ ..... 156  
*バイブライン処理とマルチスレッディング*
  - 5 二択の賭けにはどっちにも張る ..... 160  
*投機的実行*
  - 6 進捗会議はチャットで  
同時書き込みさせろ ..... 165  
*マルチスレッディング式会議*
  - 7 褒める叱るも合理的にやれ ..... 169  
*強化学習*
  - 8 8時間に6つ製品を作らせてみる ..... 173  
*ハッカソンと遺伝的アルゴリズム*
  - 9 なぜ行動基準がわかりやすいリーダーが  
成功するのか? ..... 178  
*Googleの人事考課*
- COLUMN 4 ルームシェアと負荷分散 ..... 184

# 5

## 視野を広げて ビジネスを設計する

1	お金を多次元で捉える	189
	<i>ほかし処理とスケラビリティ</i>	
2	潔く選択肢を捨てる	195
	<i>トレードオフ</i>	
3	取締役には2種類ある	198
	<i>フレームワークとライブラリ</i>	
4	ビジネスの構造を理解・設計する	202
	<i>アーキテクト</i>	
5	プログラマーは未来を予見する	210
	<i>イノベーション</i>	
COLUMN 5	ハックが最適化のすべてである	218
	あとがき	222