SecHack365

# タイムラインを活用したプログラムの動作を可視化する学習システムの開発

開発駆動コース　川合ゼミ　34Dk 丸山拓真

## 概要

**高水準言語の処理の流れを可視化するソフトウェア**

**可視化のための専用言語も開発！**

本システムは、高水準プログラミング言語の流れを可視化するツールです。解析対象プログラムに対して、コンパイラ理論を用いた解析を行い、プログラムの流れをトレースします。（図1）トレースした結果を「タイムライン」というUIを用いて可視化します。UMLはオブジェクト単位なのに対し、本システムはコード1行単位で可視化します。（図2）プログラミング初学者の理解との助けとなると同時に、デバッカとしての活用を担うことを目標に、開発を進めています。

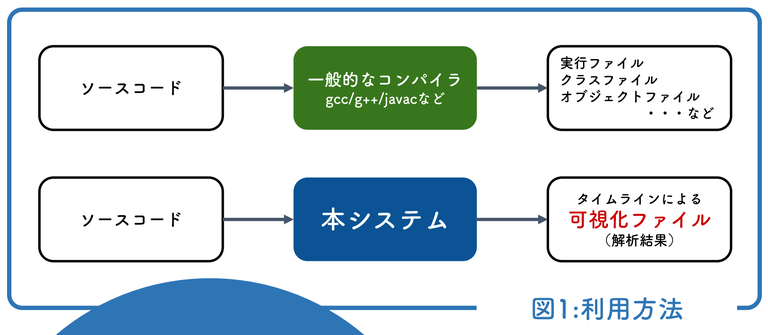


図1:利用方法

## 可視化

### タイムラインUIを活用

動画編集ソフト等で広く採用されている、タイムラインUIを活用。タイムラインUIは、「時間」という軸で、コンピューターと人間をシームレスに接続するもの。この性質を持つタイムラインを活用することで、プログラムの流れを分かりやすく可視化します。

タイムライン

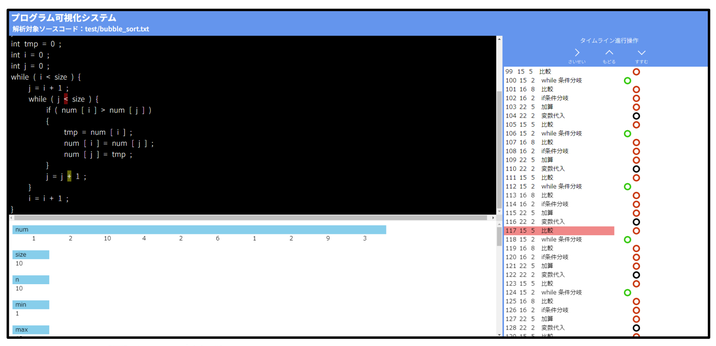


図2:可視化画面

解析によって出力される可視化ファイル(HTML)の表示

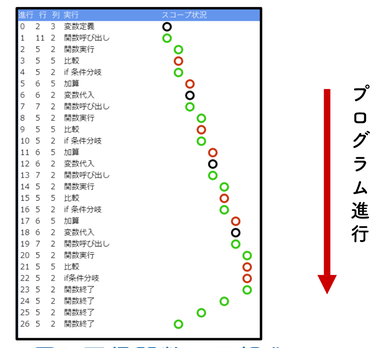


図3:再帰関数の可視化

再帰関数の可視化結果の表示

## 解決する問題

### プログラミング初学者だけど、プログラムの流れがつかめないよ

本システムの開発動機は、友達の「プログラミングの流れがつかめないよ」という悩みです。確かに、プログラミングを始めたてだと、if文とかfor文などが恐ろしく見えてしまうかもしれません。そのような悩みを取り除くことと合わせて、複雑なアルゴリズムへの入り口としても活用できるようにしたいと考えています。

### 再帰関数を使ったアルゴリズムがわからない アルゴリズムの流れがつかめないよ

再帰関数を使ったアルゴリズムはたくさんあります。木構造やグラフなどのデータ構造を探索するときに、ダイクストラ法や幅優先探索、深さ優先探索などのアルゴリズムが大いに役立ちます。それ以外でも、フィボナッチ数列の計算など、さまざまな場面で活躍する再帰関数ですが、再帰関数そのものの性質から、トレースしにくいということを耳にします。そんな時に、本システムです。本システムは再帰関数にも対応しており、その実行をタイムライン上で可視化することができます。（図3）再帰関数が怖いとはもう言わせません！

### プログラムが複雑すぎて、プログラムの流れがつかめないよ

教育目的だけでなく、強力なデバッカとしても活用できます。「複雑なアルゴリズムを書いたけれど、本当に書いたコードが動いてくれるか不安」、「うまく動かない！流れを確認してエラーをなくしたい！」というときにも役立ちます。眠たくてプログラムの流れがつかめないときも、このツールを使えばすぐに解決します。

### 符号理論・暗号理論のアルゴリズムがわからない アルゴリズムの流れがつかめないよ

セキュリティ分野においては、さまざまなアルゴリズムが活用されており、そのアルゴリズムの理解は、とても大切なことです。本システムでは、内部に存在する専用言語の言語仕様が許す限り、さまざまなアルゴリズムを可視化することができます。すでに符号理論のアルゴリズム、「ランレングス圧縮」アルゴリズムについての可視化に成功しています。本システムは、セキュリティ教育についても、アルゴリズム面から貢献することができます。

言語処理

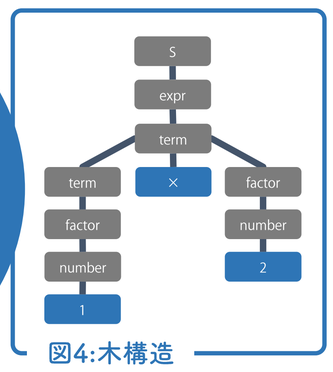


図4:木構造

### プログラミングの授業で 解説資料を作るのがめんどくさい！

本システムは、高水準言語の可視化をすることができ、その可視化結果が記載されている可視化ファイルは、単一HTMLファイルで出力されます。単一HTMLで出力されますから、その扱いは容易で、Google DriveやGoogle Classroomに簡単にアルゴリズム解説資料としてアップロードすることもできます。もう、プログラムの動作やアルゴリズムの動きを説明する資料を、プレゼンテーションソフトウェアで、解説資料を頑張って作る必要もありません。本システムで、一発で作成することが、できるようになります。

評価

## 解析

### プログラムの流れを解析するためコンパイラの仕組みを活用

プログラムの流れを可視化するためには、「どういった形でif文が使われているのか、四則演算をしているのか、代入操作が行われているのか」というような、構文の組み合わせを解析する必要があります。本システムでは、この解析にLR(1)法と呼ばれる解析手法を採用しました。解析対象となるソースコードに対して、LR(1)法に基づいた構文解析を行い、図4のような、木構造を生成します。生成された木構造を走査することで、プログラムの流れを知ることができるのです。

### 専用言語は文法定義を容易に変更できる！

本システムでは、学習者の「C言語の学習がしたい」「Java言語の学習がしたい」などのような要求に対応するため、柔軟に学習対象言語を模した言語を作成できる仕組みが必要になりました。学習対象言語を模した言語を作成するにあたっても、学習者を管轄する担当教員等の負担にならない程度に容易に作成できる仕組みや、言語仕様プリセットとして有志が容易に配布できる体制の整備も必要になってきます。つまるところ、解析対象とする専門言語の言語仕様の柔軟な文法定義が不可欠なのです。この要求を満たすために、柔軟な文法定義を行える仕組みを作りました。BNF文法定義から構文解析テーブル生成モジュールを通して生成される構文解析テーブルファイルと、構文木走査モジュールを差し替えることによって、文法定義を容易に変更することができます。

## フィードバック

### フィードバックを実施！

解析の結果出力される可視化ファイルについて、フィードバックを実施しました。
フィードバックの実施方法は、可視化ファイル（バブルソート・ランレングス圧縮の可視化を行ったファイル）を配布し、それを閲覧してもらい、Googleフォームにて評価を受け付ける方式で行いました。５名の方から評価をいただきました。
フィードバックの結果について、抜粋して紹介します。

#### フィードバックコメント

- プログラムがどう動いているのか(How)の部分についてグラフィカルに表現され、視覚的に非常に面白く、抽象的な概念に対する親しさを覚える
- 初心者から中級者以上まで幅広く役立てられそうなので、このプロダクトの秘めるポテンシャルはかなり大きい
- 再帰関数を使うプログラムなどは頭の中だけでの想像に限界があるのでこういうツールがあると学習が捗る
- タイムラインに表示する粒度を式単位ではなく行単位、ブロック単位にできると見やすさがより向上するかも
- 配列のswapなど重要な処理に関して、アニメーションでその様子が強調されたら教材レベルになりそう

**成果物公開情報**

プロジェクトページ
https://shio3001.github.io/SoftLoopWebsite/

NICT National Institute of Information and Communications Technology