

セキュアで楽に書けて速いプログラミング言語 NEK-OT

開発駆動コース 川合ゼミ 秋山 陸

テーマの概要

他のどの言語も一長一短。構文が理解しづらい言語も存在する。本プロジェクトは、競技プログラミングでの利用を想定し、開発する。言語特徴として、直感的に理解でき、**静的型付け&メモリ安全性**というコンセプトのもと、プログラミング言語を開発する。

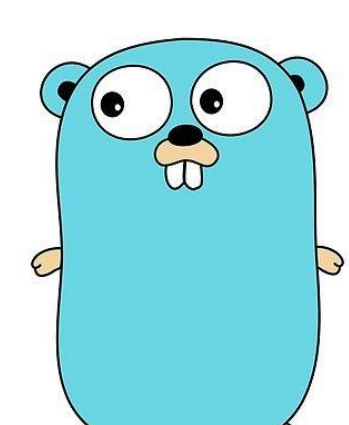
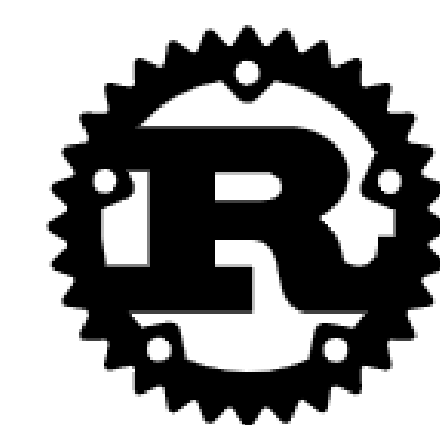
LLVMとは

- ・コンパイラ基盤
- ・コンパイル時、リンク時、実行時などあらゆる時点で**最適化**をする
- ・**実行速度が速い**



LLVMが使われている言語&コンパイラ

- ・Clang(C++)、Rust、Goなどの多くの言語で利用



nek-ot

内部

nek-ot(ネコット)と読む。

対象

- ・楽に書けるかつセキュアで実行速度が速い言語を利用したい方

言語特徴

- ・**静的型付け言語**で、**競技プログラミング**に特化、実行速度が速い
- ・型推論により型を書かなくてよい
- ・エラーメッセージが分かりやすい
- ・オーバーフロー検出など、**メモリ安全性**をもつ
- ・構文が分かりやすく、可読性が高い

構文集

```
1|import "io";
2|import "compe";
3|
4|//関数定義 main関数から始まる
5|fn main() -> i32{
6|    //変数定義
7|    w := 10;
8|    h := i32 11;
9|    ary := {1, 2, 3}; //配列
10|    li := [1, 2, 3]; //リスト
11|    //ifは式を返すことも可能
12|    b := if(w = 10) true; else false;
13|    if(b) {
14|        //macro for(i in 0..10-1)と同等
15|        rep!(0, 10)
16|        writeln("Hello");
17|    }
18|    ret 0;
19|}
```

オーバーフロー検出

```
1|import "io";
2|
3|fn main() -> i32{
4|    //i32型(int)
5|    x := 10;
6|    //オーバーフロー
7|    for(i in 1..2147483647; i++){
8|        x+=i;
9|        writeln("%d", x);
10|    }
11|}
```

コンパイル&実行

Runtime Error: Overflow occurred!
at Test\err\runtime_sadd_overflow.nk : 8 : 4

配列の範囲外アクセス

```
1|import "io";
2|
3|fn main() -> i32{
4|    arr := i32[5]{1, 2, 3, 4, 5};
5|    i := 100;
6|    arr[i] = 0;
7|    arr[10] = 0;
8|    ret;
9|}
```

コンパイル

Error: Compile error
-->
Test\err\cmpltime_array_index_outofrange.nk:7:5

6| arr[i] = 0;
7| arr[10] = 0;
 ^
Array index out of range!

標準ライブラリ

- ・Algo : ソート, 順列, 組み合わせの数, 最大公約数など
- ・Compe : repなどのよく使うマクロをまとめたライブラリ
- ・Math : min, max, log, pow, PIなど

実行ファイル生成の流れ

自作言語

ソースファイル ↓

コンパイル

LLVMビットコード ↓

LLVM optimizer

LLVMビットコード ↓

リンカ

a.exe ↓

実行ファイル生成

- ・C++で実装
- ・現在約5200行

実行速度&コンパイル速度

マンデルブロ集合の計算(64bit)	Version	平均 (100回)
nek-ot -O3	-	821.4546
nek-ot -O3 unsafe	-	749.2345
GCC -O3	8.1.0	772.7747
Clang -O3	8.0.0	743.9410

マンデルブロ集合の計算のコンパイル速度(64bit)	Version	平均(100回)
nek-ot -O3	-	757.3650
GCC -O3	8.1.0	377.6926
Clang -O3	8.0.0	907.6194