中国科学技术大学
University of Science and Technology of China

# TIP Language

## TIP: Tiny Imperative Programming language


张昱

yuzhang@ustc.edu.cn

中国科学技术大学

计算机科学与技术学院

# TIP and its Implementation

中国科学技术大学
University of Science and Technology of China

## TIP language

- Minimal C-style syntax
- Enough features to make static analysis challenging and fun

## Implementation

- Scala: https://github.com/cs-au-dk/TIP/
- C++ 17: https://github.com/matthewbdwyer/tipc

# Expresions in TIP



$$
\begin{aligned}
Exp \rightarrow\ & Int \\
\mid\ & Id \\
\mid\ & Exp + Exp \mid Exp - Exp \mid Exp * Exp \mid Exp \,/\, Exp \\
\mid\ & Exp > Exp \mid Exp == Exp \\
\mid\ & (\ Exp\ ) \\
\mid\ & \texttt{input}
\end{aligned}
$$

- I∈ Int represents an integer literal
- X∈ Id represents an identifier (x, y, z, ...)
- input reads an integer from the input stream
- Comparison operators yield 0(false) or 1(true)

# Statements in TIP



```
Stm → Id = Exp;
    | output Exp;
    | Stm Stm
    |
    | if (Exp) { Stm } [else { Stm }]?
    | while (Exp) { Stm }
```

- In conditions, 0 is false, all other values are true
- The output statement writes an integer value to the output stream

# Functions in TIP



```
Fun → Id ( Id, ..., Id ) {
        [var Id, ..., Id;]?
        Stm
        return Exp;
      }
```

- The optional var block declares a collection of uninitialized variables
- Function calls are an extra kind of expressions:

```
Exp → ... | Id ( Exp, ..., Exp )
```

# Pointers



```
Exp → ...
    | alloc Exp
    | &Id
    | *Exp
    | null
```

```
Stm → ... | *Exp = Exp;
```

- No pointer arithmetic

# Records

$$
\begin{aligned}
&Exp \rightarrow \ldots \\
&\quad \mid \{\ Id : Exp\ ,\ \ldots\ ,\ Id : Exp\ \} \\
&\quad \mid\ Exp\ .\ Id
\end{aligned}
$$

$$
\begin{aligned}
&Stm \rightarrow \ldots \\
&\quad \mid\ Id\ .\ Id = Exp\ ; \\
&\quad \mid\ (*Exp)\ .\ Id = Exp\ ;
\end{aligned}
$$

- Records are passed by value (like structs in C)
- For simplicity, values of record fields cannot be records

# Functions as Values



- Functions are first-class values
- The name of a function is like a variable that refers to that function
- Generalized function calls

$$Exp \rightarrow \ldots \mid Exp\,(\,Exp\,,\,\ldots\,,\,Exp\,)$$

- Function values suffice to illustrate the main challenges with methods (in OO languages) and higher-order functions (in functional languages)

# Programs



$$Prog \rightarrow Fun \ldots Fun$$

- **A program is a collection of functions**
- **The function named main initiates execution**
  - Its arguments are taken from the input stream
  - Its result is placed on the output stream
- **We assume that all declared identifiers are unique**

# TIP Examples



## Recursive factorial function

```
rec(n) {
  var f;
  if (n==0) {
    f=1;
  } else {
    f=n*rec(n-1);
  }
  return f;
}
```

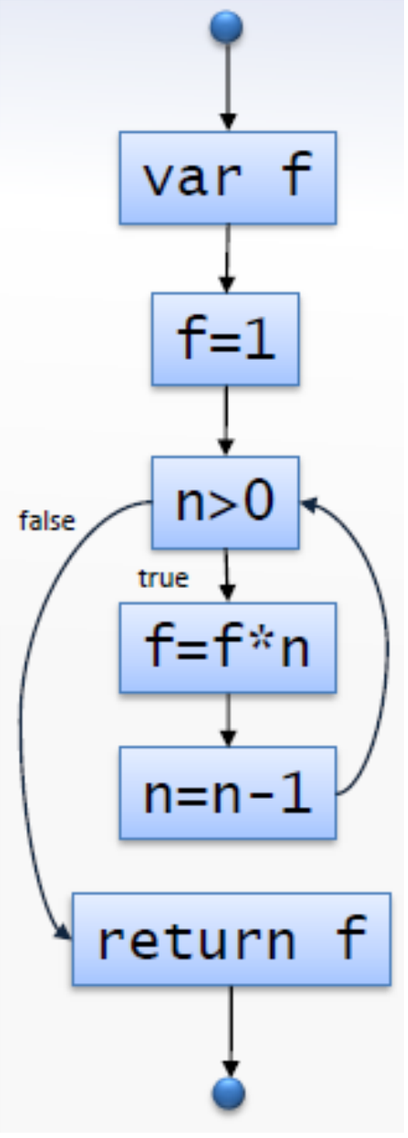
## Iterative factorial function

```
ite(n) {
  var f;
  f = 1;
  while (n>0) {
    f = f*n;
    n = n-1;
  }
  return f;
}
```

# Control flow graphs

## Iterative factorial function

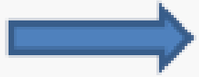
```
ite(n) {
  var f;
  f = 1;
  while (n>0) {
    f = f*n;
    n = n-1;
  }
  return f;
}
```



# Normalization

- **Normalization：flatten nested expressions, using fresh variables**

```
x = f(y+3)*5;
```



```
t1 = y+3;
t2 = f(t1);
x = t2*5;
```

中国科学技术大学
University of Science and Technology of China

# Thanks