Linguagem Imperativa 2
Tipada

Prof. Vander Alves

Linguagem Imperativa 2 ***Tipada***

- Um programa é uma lista de definição de funções
- A execução do programa começa pela função **`main`**
- LI2T = LI 2 + tipos
- Tipo: conjunto de valores com mesma estrutura e conjunto de operações
  - bool, int, String

Linguagem Imperativa 2 ***Tipada***

- Vantagens
  - Detectam alguns tipos de erros (p.ex. **`1 + true`**)
  - Documentação e organização dos programas
  - Manutenção
- Desvantagens
  - Esforço de documentação (*no free lunch!*)
  - Custo computacional (tratável)

Linguagem Imperativa 2 ***Tipada***

- Expressões e funções tem tipos
  - Tipos são determinados pela *inferência de tipos*
- Programas são *bem tipados (válidos)* ou *mal tipados (inválidos)*
  - Verificado pelo checador de tipos (***typechecker***)
- Checagem automática
    - expressões com tipo esperado em *comandos*
    - expressões com tipo esperado em *expressões*

Linguagem Imperativa 2 ***Tipada***

- A checagem de tipos é estática
  - Estaticamente: Haskell, Java, ML, C,C++, etc.
  - dinamicamente: Python, JavaScript, etc.
- Vantagens da checagem estática
  - Simplicidade
  - Efetividade e eficiência da depuração
  - Antes da distribuição e execução

Exemplos (válido/bem-tipado)

```
int main () {
 int x;
 x = 2;
 return x;
}
```

Exemplo (*inválido/mal-tipado*)

```
int main () {
    x = "Hello" + 2;
    return x;
}
```

Exemplos (inválido/mal-tipado)

```
String main () {
  int x;
  x = 20;
  x = 1 + "hello";
  String w;
  w = "hello";
  String v;
  v= "world";
  String a;
  a = w ++ 20;
  a = w ++ v;
  return a;
 }
```

Exemplos (válido/bem-tipado)

```
String main () {
  int x;
  x = 20;
//  x = 1 + "hello";
  String w;
  w = "hello";
  String v;
  v= "world";
  String a;
//  a = w ++ 20;
  a = w ++ v;
  return a;
 }
```

Exemplo (válido/bem-tipado)

```
int main () {
   int f;
   f = fat (5);
   return f;
}

int fat (int n) {
 int r;
 if (n)
     then r = n * fat (n-1);
     else r = 1;
 return r;
}
```

Exemplo (inválido/mal-tipado)

```
int main () {
   int f;
   f = fat (5);
   return f;
}

int fat (int n) {

 if (n)
    then r = n * fat (n-1);
    else r = 1;
 return r;
}
```

Exemplo (válido/bem-tipado)

```
int main () {
  int f;
  f = fat (5);
  return f;
}

int fat (int n) {
 int r;
 if (n)
    then r = mult (n,fat (n-1));
    else r = 1;
 return r;
}

int mult (int a, int b) {
  int r;
  r = a * b;
  return r;
}
```

Exemplo (inválido/mal-tipado)

```
int main () {
  int i;
  int j;
  int k;
  j = 100;
  k = 20;
  i = soma (j,k);
  return i ;
}

int soma (int a, int b) {
  int r;
  r = a + b + j;
  return r;
}
```

Exemplo (válido/bem-tipado)

```
int main () {
 int x;
 int y;
 int z;
 int w;
 x = 3;
 y = 4;
 z = fSoma(x);
 w = fSoma(y);
 return z+w;
}

int fSoma(int n) {
 int soma;
 soma = 0;
 int c;
 c = n;
 while (c) {
   soma = soma + c;
   c = c - 1;
  }
 return soma;
}
```

Exemplo (válido/bem-tipado)

```
int main () {

  int i;
  i = 2;

  {
    int i;
    i = 3;
  }

  return i;

}
```

Exemplo (inválido/mal-tipado)

```
int main () {

  int j;
  j = 2;

  {
    int i;
    i = 3;
  }

  j = i;
  return j;

}
```

Exemplo (válido/bem-tipado)

```
int main () {

  int j;
  j = 2;

  {
    int i;
    i = 3;
    j = i;
  }

  return j;

}
```

Exemplo (válido/bem-tipado)

```
void main () {
  int z;

  int x;
  x = 1;

  int y;
  y = 0;

  {
    x = 2;
    y = x;
  }

  z = x;
}
```

Exemplo (válido/bem-tipado)

```
void main () {
  int x ;
  String w;

    {
      x = 3 ; // x : int
      String x ; // x : String
      x = "hello world" ;
      int z ;
      //z = x + 1;
      w = x;
    }
  x = x + 1 ; // x : int, recebe o valor 3 + 1
  //z = 8 ; // ILEGAL! z nao esta mais no escopo
  //bool x ; //ILEGAL!x nao pode ser redeclarada
}
```

Projeto e Implementação

- Sintaxe concreta
- Sintaxe abstrata
- Checador de tipos (*typechecker*)
- Interpretador
- Vide implementação no arquivo no Moodle:

  `aulas\93_LITipada\LI2Tipada.rar`