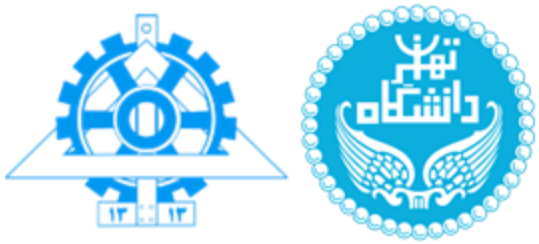

به نام خدا

تمرین کامپیوتری سوم درس طراحی کامپایلر

بهار ۱۴۰۳

مهلت تحویل:1403/3/4

طراحان : محمدجواد پسرکلو، محمدرضا نعمتی

فهرست مطالب

مقدمه

در فاز دوم پروژه، خطاهای مربوط به اسامی توابع، متغیرها و pattern ها بررسی شد. در طراحی یک کامپایلر، فاز دیگر Semantic analysis، بررسی خطاهای مربوط به type است. در این فاز از پروژه‌ی درس قصد داریم تعدادی از type error هایی که ممکن است در زبان FunctionCraft رخ دهد را بررسی کنیم.

یکی از مهم‌ترین کارها در type checking زبان‌های functional، infer کردن type توابع است. چرا که در بسیاری از زبان‌های functional، نوع پارامترها و مقدار بازگشتی توابع به صورت ضمنی در تعریف توابع مشخص نمی‌شود. ممکن است در این زبان‌ها یک تابع به صورتی تعریف شده باشد که بتواند با ورودی‌هایی از type های مختلف فراخوانی شود. در این حالت همان‌گونه که در درس فرا گرفتید، تا قبل از فراخوانی این توابع برای آن‌ها type variable در نظر گرفته می‌شود.

برای جلوگیری از پیچیدگی‌های type variable، از شما انتظار داریم توابع را با دنبال کردن flow برنامه از تابع main، بررسی کنید و type checking را انجام دهید. با این روش، توابعی که در هیچ قسمت از برنامه فراخوانی نشده‌اند، type check نمی‌شوند. پس در این تمرین، type checking هر تابع فقط **هنگام فراخوانی** و **به ازای هر فراخوانی آن** انجام می‌شود.

برای انجام پروژه، متدهای visit از TypeChecker Visitor را تکمیل کنید و در صورت تشخیص خطا، یک instance جدید از خطای مربوطه به آرایه *typeErrors* اضافه کنید. قابل ذکر است که برای آشنایی بیشتر شما، تعدادی از خطاهای دیگر در کد handle شده‌اند.

حذفیات

تعدادی از feature های زبان برای فازهای باقی‌مانده (3 و 4) حذف شده‌اند و نیازی به پیاده‌سازی هیچ قسمتی از visitor های آن‌ها نیست:

1. **Lambda function**
2. **elseif**
3. **two or multi-dimensional lists**

خطاهای type inference

خطاهایی که انتظار می‌رود در این فاز پیاده‌سازی کنید به شرح زیر است:

1. یکسان نبودن type عملوندهای مربوط به یک عملگر (NonSameOperands)

عملگرها ممکن است نیاز به عملوندهایی با type های یکسان داشته باشند.

```
def main()
    a = 1 + "aaa";
end
```

```
Line:2-> operands of operator PLUS must be the same
```

2. مجاز نبودن عملوند برای یک عملگر (UnsupportedOperandType)

عملگرها ممکن است است مربوط به یک یا چند type مشخص باشند و برای بقیه type ها امکان استفاده نداشته باشند.

```
def main()
    b = "aa" + "aa";
end
```

```
Line:2-> unsupported operand type for operator PLUS
```

3. یکسان نبودن type مقادیر بازگشتی یک تابع (FunctionInconsistentReturnTypes)

مقادیر return شده توسط یک تابع باید type های یکسان داشته باشند.

```
def func(a, b, [c = 10])
    if (!(c == 10))
        return 10;
    else
        return 20.1;
    end
    return "test";
end


def main()
    func("a", 1);
end
```

```
Line:1-> types of return expressions of the function `func` mu
st be same
```

4. یکسان نبودن type مقادیر موجود در لیست (ListElementsInconsistentTypes)

type عناصر موجود یک لیست باید یکسان باشد.

```
def main()
    c = [1, 2, 3, "string", true];
    for i in [1, 2, "string2", true]
        puts(i);
    end
end
```

```
Line:2-> all elements of the list must have same type
Line:3-> all elements of the list must have same type
```

5. صحیح نبودن type اندیس (AccessIndexIsNotInt)

index ها باید از نوع integer باشند.

```
def main()
    c = [1, 2, 3, 4];
    d = c["index"];
end
```

```
Line:3-> access index must be integer
```

6. صحیح نبودن type ورودی تابع len (LenArgumentTypeMisMatch)

تابع len فقط ورودی با type لیست یا string می‌تواند دریافت کند.

```
def main()
    e = 22.2;
    f = len(e);
end
```

```
Line:3-> type of `len` function argument must be list or strin
g

```

7. صحیح نبودن type ورودی‌های تابع push (PushArgumentsTypeMisMatch)

ورودی‌های تابع push یا هر دو باید از نوع string باشند و یا اینکه type مقدار push شونده، با type عناصر لیست یکسان باشد. همچنین ممکن است لیست خالی باشد که در این صورت پس از push شدن یک عنصر به لیست، type عناصر بعدی لیست باید با type اولین عنصر وارد شونده در آن یکسان باشد.

```
def main()
    push([1, 2], "string");
    push("string", 1);
end
```

```
Line:2-> types of `push` function arguments must be string
or the type of the second argument must be the same as the type of the elements in the list
Line:3-> types of `push` function arguments must be string
or the type of the second argument must be the same as the type of the elements in the list
```

8. مجاز نبودن انجام push (CannotBePushed)

فقط به string ها و لیست‌ها امکان push شدن وجود دارد و بقیه type ها نمی‌توانند به عنوان آرگومان اول تابع push قرار بگیرند.

```
def main()
    push(2, 1);
    push(22.2, "string");
end
```

```
Line:2-> only lists and strings can get pushed into
Line:3-> only lists and strings can get pushed into
```

نکات مهم

- تمامی فایل‌ها و کدهای خود را در یک فایل فشرده به صورت studentID1_studentID2.zip آپلود نمایید.
- در صورت کشف هرگونه تقلب، نمره صفر لحاظ می‌شود.
- دقت کنید که خروجی‌ها به صورت خودکار تست می‌شوند؛ پس نحوه چاپ خروجی باید عیناً مطابق موارد ذکر شده در بالا باشد.
- بهتر است سوالات خود را در فروم درس یا در گروه اسکایپ مطرح نمایید تا دوستانتان نیز از آنها استفاده کنند؛ در غیر این صورت به مسئولان پروژه ایمیل بزنید.