سند زبان برنامه‌نویسی

Jepeto

نسخه 1...1

فهرست مطالب

1. مقدمه

زبان ابداعی Jepeto یک زبان برنامه نویسی تابعی مشابه زبان ML است که در آن type declaration وجود ندارد و از Type Inference استفاده می شود. در این زبان دستور assignment، ساختار تکرار و دستورات حلقه نداریم. این زبان قابلیت خاص anonymous function را دارد که در ادامه بیشتر توضیح داده می‌شود. این زبان از انواع داده‌ای پیش‌تعریف bool، int و string استفاده می کند و همچنین قابلیت تعریف لیست از یکی از انواع گفته شده را دارد. در هر فایل از این زبان، یک بخش main داریم که مشخص می‌کند برای اجرای برنامه چه تابعی با چه مقادیر اولیه‌ای فراخوانی شود.

2. ساختار کلی

در این زبان، کد برنامه درون یک فایل با پسوند jp. قرار دارد. این فایل از قسمت های زیر تشکیل شده است:

- تعریف تعدادی تابع
- یک بخش main

یک نمونه از کد در این زبان به صورت زیر است:

```
func testMethod(test1, test2, i):
    if i not 10:
       return testMethod (test1, test2::i, i+1);

    else:
```

```
        if test1 is test2[i]:
            return true;
        else:
            return false;

main: print(testMethod(10,[],0));
```

2-1. قواعد کلی نحو

زبان Jepeto به بزرگ و کوچک بودن حروف حساس است. در این زبان، وجود کاراکترهای Tab ، Space و خط جدید تاثیری در خروجی برنامه ندارند. جزئیات مربوط به Scope ها و خطوط برنامه در ادامه به طور مفصل توضیح داده خواهد شد.

2-2. کامنت‌ها

در این زبان، کامنت‌ها تنها تک‌خطی هستند و تمامی کاراکترهای بعد از # تا انتهای خط کامنت به حساب می‌آیند.

```
func getLastElement(L):
       return L[L.size-1];

main: print ((getLastElement([“A”,”B”,”C”]));
    # This is a comment!
```

2-3. قواعد نام‌گذاری توابع و آرگومان‌ها

اسامی انتخابی برای نام‌گذاری توابع و آرگومانها باید از قواعد زیر پیروی کنند:

- تنها از کاراکترهای a..z،A..Z، _ و ارقام تشکیل شده باشند.
- با رقم شروع نشوند.
- معادل کلیدواژه‌ها نباشند. در جدول زیر تمامی کلیدواژه‌های زبان Jepeto آمده‌است:

bool	string	int	and	or
is	not	true	false	if
else	func	main	return	void
print	size			

- نام هر تابع، یکتاست.
- نام هر آرگومان در یک Scope یکتاست.

3. بخش main و نحوه فراخوانی توابع

بخش main به صورت زیر تعریف می شود:

```
main :  method(); # a function call
```

تنها یک فراخوانی در تابع main است و در آن توالی‌ای از فراخوانی‌های توابع مختلف ممکن نیست. هم چنین بدنه main می تواند از یک گزاره print (که یک تابع پیش‌تعریف است) تشکیل شده باشد. main مقدار بازگشتی ندارد.

تعریف یک تابع به صورت زیر انجام می‌شود:

```
func method(arg1, arg2, arg3, ...): {
      # body
      return expression;
}
```

همه توابع باید مقدار بازگشتی داشته باشند. اگر تابعی مقدار بازگشتی نداشته باشد، صریحا کلیدواژه void را بر می‌گرداند (return void;).

فراخوانی متدها یک Statement و Statement دیگر این زبان return است. در انتهای Statement یک کاراکتر ; نوشته می‌شود. برای تعریف بلاک از {} استفاده می شود (اگر چند دستور متوالی داشته باشد باید در یک بلاک قرار گیرند). همچنین در این زبان، فراخوانی توابع به صورت call-by-reference است.

4. توابع گمنام یا anonymous function

در این زبان یک نوع تابع خاص به نام توابع گمنام داریم. این توابع بدون اسم گذاری و در زمان فراخوانی یک تابع تعریف می‌شوند و تنها در همان تابع، قابل فراخوانی هستند. به طور مثال در کد زیر maxfinder یک تابع معمولی و equal یک تابع گمنام است که تنها در تابع maxfinder قابل استفاده است.

```
func maxfinder(arr,i,equal,max):
     if arr.size < i:
         return max;

     else :
      return maxfinder(arr,i+1,equal,equal(arr[i],arr[i+1]));

main: maxfinder([1,2,3,4,5], 0, (a,b)->{if a>b: return a; else: return b;}, -1);

# It means that : equal = (a,b)->{if a>b: return a; else: return b;}
```

5. انواع داده

در زبان Jepeto، سه نوع پایه int، string و bool و دو نوع دیگر list و fptr که توضیحات آنها در ادامه آمده است، وجود دارند. در این زبان، انواع int، string و bool از نوع primitive هستند.(خود مقادیر در آن‌ها ذخیره می‌شوند نه اشاره‌گری به خانه‌ای از حافظه) و انواع دیگر، از نوع non-primitive هستند و در آن‌ها اشاره‌گری به خانه‌ای از حافظه وجود دارد.

5-1. لیست

لیست دارای تعدادی عناصر است در بین نماد [ ] می آیند و با کاما جدا می شوند. در این زبان، اعضای لیست باید حتما از یک نوع باشند.

نوع عناصر لیست می‌تواند یکی از چهار نوع string، bool، int و list باشد.

مثال :

[1, 2, 3] ✔ [1,2,"hi"] ❌ [true, "hi"] ❌ [[1,2,3], [“hi”, “bye”]] ❌

[[“hello”,”world“,”PLC“], [“hi”, “bye”]] ✔ [1, 2, [2, 3, 4]] ❌

[[[1, 2], [2, 2], [3, 5]], [[4], [4, 5]], [[9]]] ✔

[[[1, 2], [2, 2], [3, 5]], [[4], [4, 5]], [9]] ❌ : عضو اول و دوم لیستی از لیست هاست

اما عضو سوم لیستی از int میباشد . پس نوع اعضا یکسان نیست.

5-2. اشاره گر به توابع

توابع گمنام و نام توابع وقتی به آرگومان توابع پاس داده می‌شوند یا به عنوان مقدار بازگشتی تابعی دیگر تعیین می‌شوند از نوع fptr هستند. در نتیجه عمل صدا زدن تابع تنها روی نوع fptr صورت می‌گیرد.

در مثال زیر هر دو آرگومان f1 و f2 از نوع fptr هستند و همچنین در تابع sum میتوان مشاهده کرد که مستقیما روی anonymous function می توانیم فراخوانی داشته باشیم.

```
main : print(sum(g, (a) -> {return a*2;}));

func sum(f1, f2) : {
   return f1() + f2(2) + (num) -> {return num / 5;}(20);
}

func g() :
   return 8;
```

6. آرگومان‌ها

آرگومان‌های تابع می‌تواند از صفر تا چند مورد و از هر یک از انواع پیش‌تعریف و یا لیست و یا اشاره گر به تابع باشد.با فراخوانی یک تابع با مقادیر ورودی مشخص، آرگومان‌های آن را مقداردهی می‌کنیم. هنگام فراخوانی، آرگومان ها به دو صورت می توانند پاس داده شوند: یا عبارت به عنوان آرگومان پاس داده می شود یا از ترکیب نام آرگومان در تعریف تابع و عبارت مورد نظر استفاده می شود که در این حالت الزامی نیست ترتیب ورودی آرگومان ها با ترتیب آنها در تابع یکسان باشد. در ضمن یا همه آرگومان‌ها به صورت عبارت‌اند و یا همه به صورت نام آرگومان و عبارت متناظر آن. به مثال های زیر توجه کنید:

فراخوانی با روش اول:

```
main : f(10, true);

func f(a, b) : {
  if (b) :
      print(a);
  return void;
}
```

فراخوانی با روش دوم:

```
main : f(b=true, a=10);

func f(a, b) : {
  if (b) :
      print(a);
  return void;
}
```

7. عملگرها

عملگرها در زبان Jepeto به پنج دسته‌ی عملگرهای حسابی، مقایسه‌ای، منطقی، عملگر الحاق، عملگر سایز و عملگر تخصیص تقسیم می‌شوند.

7-1.عملگرهای حسابی

این دسته از عملگرها تنها روی اعداد عمل می‌کنند، لیست این عملگرها در جدول زیر آمده است. در مثال‌های استفاده شده A برابر 20 و B را برابر 10 در نظر بگیرید:

عملگر	شرکت‌پذیری	توضیح	مثال
+	چپ	جمع	A + B = 30
-	چپ	تفریق	A - B = 10

*	چپ	ضرب	A * B = 200
/	چپ	تقسیم	A / B = 2 B / A = 0
-	راست	منفی تک‌عمل‌وندی	-A = -20

7-2. عملگرهای مقایسه‌ای

این عملگرها وظیفه‌ی مقایسه را دارند، پس نتیجه‌ی آن‌ها باید مقدار صحیح یا غلط (true, false) باشد. با این حساب خروجی این عملگرها یک bool است. توجه داشته باشید که عملوند عملگرهای < و > تنها از جنس عدد صحیح هستند. همچنین برای عملگرهای is و not نیز باید نوع عملوندها یکسان و حتما از انواع primitive باشند؛ در غیر اینصورت باید خطای کامپایل گرفته شود.

لیست عملگرهای مقایسه‌ای در جدول زیر آمده است. در مثال‌های استفاده شده مقدار A را برابر 20 و مقدار B را برابر 10 بگیرید:

عملگر	شرکت‌پذیری	توضیح	مثال
is	چپ	تساوی	(A is B) = false
not	چپ	عدم تساوی	(A not B) = true
<	چپ	کوچکتر	(A < B) = false
>	چپ	بزرگتر	(A > B) = true

7-3. عملگرهای منطقی

در زبان Jepeto عملیات منطقی تنها روی نوع داده‌ی Bool قابل اعمال است.این عملگرها در جدول زیر لیست شده‌اند. در مثال‌های استفاده شده A را برابر true و B را برابر false در نظر بگیرید:

عملگر	شرکت‌پذیری	توضیح	مثال
and	چپ	عطف منطقی	(A and B) = false
or	چپ	فصل منطقی	(A or B) = true
~	راست	نقیض منطقی	~A = false

7-4. فیلد سایز

برای بدست آوردن تعداد اعضای یک لیست به کار می رود .

در مثال استفاده شده A را [[7 , 6 , 5] , [4 , 3] , [2 , 1]] در نظر بگیرید :

A.size = 3

A[0].size = 2

7-5. عملگر الحاق

برای اضافه کردن یک عضو به یک لیست به کار می رود. در نظر داشته باشید که نوع عضوی که قرار است اضافه شود باید از نوع دیگر عناصر لیست باشد.

در مثال استفاده شده A را [[7 , 6 , 5] , [4 , 3] , [2 , 1]] و B را [3 , 2 , 1]در نظر بگیرید :

A :: [12 , 13 , 14 , 15] => A = [[1 , 2] , [3 , 4] , [5 , 6, 7] , [12 , 13 , 14 , 15]]

B :: 5 => B = [1 , 2 , 3 , 5]

7-6. عملگر تخصیص

در زبان Jepeto نماد = را عملگر تخصیص می‌نامیم و همان طور که پیش از این توضیح داده شد برای نسبت دادن نام آرگومان های یک تابع به عبارت مورد نظر هنگام فراخوانی استفاده می‌شود.

7-7. اولویت عملگرها

اولویت عملگرها طبق جدول زیر است:

اولویت	دسته	عملگرها	شرکت‌پذیری
1	پرانتز	( )	چپ
2	براکت	[ ]	چپ
3	سایز	.size	-
4	الحاق	::	چپ
5	تک عملوندی	~ -	راست
6	ضرب و تقسیم	* /	چپ
7	جمع و تفریق	+ -	چپ
8	رابطه‌ای	< >	چپ
9	مقایسه‌ای تساوی	is not	چپ
10	عطف منطقی	and	چپ
11	فصل منطقی	or	چپ

12	تخصیص	=	چپ به راست
13	کاما(ورودی متدها)	,	چپ به راست

8. ساختار تصمیم‌گیری

در زبان Jepeto تنها ساختار تصمیم‌گیری، if... else است. همچنین ساختار if می‌تواند بدون else استفاده گردد:

```
if a is 2:
	return true;
else:
	return false;
```

9. قوانین Scopeها

9-1 Scopeهای موجود در زبان

به طور کلی در زبان Jepeto موارد زیر در اسکوپ جدیدی قرار دارند:

1- آرگومان‌ها و خطوط کد داخل یک متد.

2- خطوط کد داخل گزاره های تصمیم گیری

9-2. قوانین Scopeها

نکات زیر در مورد Scopeها وجود دارد:

- متغیرهایی که داخل یک تابع تعریف می‌شوند در Scopeهای بیرون آن دسترس‌پذیر نیستند و صرفاً در Scopeهای درون آن قابل دسترسی هستند.
- امکان تعریف آرگومان با نام یکسان در یک Scope وجود ندارد.

- در Scope هر تابع باید تمامی مسیر های احتمالی که میتوانند رخ بدهند و یا حتی رخ ندهند باید دارای دستور return باشند.
- خطوط خالی از کد اجرایی هیچ تاثیری در خروجی و اجرای برنامه ندارد.

10. توابع پیش‌فرض

در زبان Jepeto، تنها یک تابع پیش‌فرض وجود دارد و آن هم تابع print است. این تابع به صورت ضمنی تعریف شده است و می‌تواند یک لیست (با هر طولی) و یا یک مقدار int یا string و یا bool دریافت کند و آن را در خروجی چاپ کند. نمونه‌ای از این دستور به صورت زیر است:

```
print ("Hello Programmer!");
```

11. مثال ها

در ادامه، چندین کد نمونه از این زبان آمده‌است:

نمونه کد یک

```
func Multiply (a,b): {
   if a is 0:
      return 0;

   return a + Multiply(a-1,b);
}

main: print(Multiply(4,5));
```

نمونه کد دو

```
func result(i, j, l, l_size):
   if i < l_size:
       if j < l_size:
           if l[j] < l[i]:
               return 1 + result(i, j + 1, l, l.size);
           else:
               return result(i, j + 1, l, l.size);
       else:
           return result(i + 1, i + 2, l, l.size);
   else:
       return 0;

main : print(result(0, 1, [11, 19, 16, 14, 15], 5));
```

نمونه کد سه

```
main : print(getOdds([], 1, 100));

func getOdds(list, i, end) : {
   if i is end : {
       printList(list);
       return void;
   }
   getOdds(list::i, i+2, end);
   return void;
}

func printList(list, i) : {
   if size_ not list.size : {
       print(list[i]);
       printList(list, i+1);
       return void;
   }
   else :
       return void;
}
```