فاز اول پروژه اصول طراحی کامپایلر

در این پروژه قصد داریم یک سری دستورات ساده و کمی پیچیده برای یک زبان برنامه نویسی را طراحی کنیم و کامپایلر آن را با استفاده از زیرساخت LLVM بسازیم. در نهایت object code آن را با کمک LLVM نمایش میدهیم.

زبان برنامه نویسی فرضی ما دارای دستورات زیر است:

- **تعریف متغیر**

در زبان طراحی شده، متغیرها دارای مقادیری هستند که در زمان compile مشخص خواهد شد. Data type های زبان ما int, float, bool , char , string و array می باشند که در ابتدا مقدار دهی اولیه ندارند . سینتکس تعریف متغیر ها به شکل مقابل می باشد:

```
int x;
boolean flag;
float pi;
string txt;
char ch;
array arr;
```

تایپ array مجموعه ای از مقادیر است که میتواند از نوع int یا float یا bool باشد.

مثال مقداردهی اولیه:

```
int a = 10;
boolean e = true, f, d = false;
char c = 'v';
string s = "abcd";
float fl = 1.53;
array l = [1, 2, 3];
```

میتوان چند متغیر از یک تایپ را باهم تعریف کرد :

```
int a , b ,c = 10;
int a,b,c = 1,2,3 ;
bool e = true , f , d = false ;
```

برای نام گذاری متغیر ها تنها مجاز به استفاده از اعداد، حروف و "_" هستیم. همچنین نام متغیر نمی تواند با عدد شروع شود. دقت شود که کلیدواژه های موجود در زبان نمی تواند به عنوان نام یک متغیر به کار روند
مثال هایی از نام گذاری های **اشتباه**:

```
int for ;
bool 12a ;
float _12a ;
```

- **انتساب متغیر**

برای انتساب متغیر هایی از نوع int و float از عملگر های =/ , =* , =% , =+ , =- , = استفاده میکنیم و برای متغیر هایی از نوع string , char , array و bool از = مجاز به استفاده هستیم.

مقادیر مجاز برای متغیر bool : true , false و حاصل عبارت های منطقی معتبر باید استفاده شود .

```
Is_tr = (3<=z) xor false ;
```

مقادیر مجاز برای متغیرهای int , float : اعداد و حاصل عبارت های ریاضی معتبر. همچنین اعداد می توانند با علامت مثبت، منفی و یا بدون علامت باشند. اما متغیر ها را نمی توانیم به صورت

مستقیم با علامت - قرینه کنیم. برای قرینه کردن متغیر ها و عبارت ها باید پیش از پرانتز عالمت - را قرار داد. برای مثال:

```
Fl = +4 - 6 * - (z/2) ;
```

مقادیر مجاز برای char : کاراکتر های معتبر.
مقادیر مجاز برای string : مجموعه از کاراکتر ها بصورت یک رشته.

- ## پیاده سازی Unary operator

عملگرهای یونری بر روی متغیرهای از نوع int و float و عنصر آرایه عددی قابل انجام می باشند.

```
x++;
x--;
```

- ## دستورات شرطی

این دستورات شامل شرط و حلقه میباشند. برای سادگی در بدنه آنها مجاز به تعریف متغیر جدید نیستیم، اما میتوانیم آنها را به صورت تودرتو استفاده کنیم .

بلوک شرط با کلیدواژه if آغاز میشود. سپس یک عبارت منطقی درون پرانتز نوشته شده و در ادامه آن، بدنه شرط در میان { } قرار میگیرد. در صورتی که حاصل عبارت منطقی true باشد، بدنه شرط اجرا میشود و در غیر اینصورت بدنه اجرا نمیشود. میتوان به جای عبارت منطقی از یک متغیر boolنیز استفاده کرد.
پس از if میتوانیم چندین else if و یا یک else داشته باشیم .

```
if ( x == 9 ) {
     print(5) ;
} else if (m >= 3) {
     S = “ali” ;
```

```
} else {
    A -= 1;
}
```

- **دستورات حلقه**

1. for

بلوک حلقه با کلید for اغاز میشود و سینتکس داخل ان به شکل زیر خواهد بود .

```
for ( int i = 0; i<10; i++) {
       a++;
}
```

2. foreach

این حلقه برای پیشمایش ارایه عناصر آرایه ها بکار می رود.

```
foreach (num in numbers) {
       print(num);
}
```

```
array values = [1, 2, 3, 4, 5];
 foreach (val in values){
     val += 5;
     print(val);
 }
```

- **کامنت**

در میان دستورات امکان کامنت گذاشتن وجود دارد. سینتکس آن به شکل زیر است:

```
/* Comment */

/* Comment1
    Comment2 */
```

- **پرینت**

پرینت کردن پاسخ عبارتهای ریاضی، منطقی و متغیرها

```
print(2*5)
print(s)
```

- **تابع String**

concat : این تابع در ورودی دو string گرفته و آنها را هم وصل میکند و در خروجی یک رشته از ترکیب آنها می دهد .

```
concat(str1, str2)
```

- **توابع ریاضی**

pow: این تابع دو مقدار عددی int را در ورودی می گیرد و در خروجی اولی به توان دومی را محاسبه میکند و به شکل int خروجی می دهد .

```
pow(x , y)   //x ^ y
```

abs: این تابع در ورودی یک مقدار عددی از نوع int یا float میگیرد و در خروجی قدر مطلق ان را از هر تایپی که بوده بر میگرداند

```
abs(-5)     // return 5
```

- **توابع کار با ارایه**

Length: در ورودی اش یک آرایه می گیرد و طول آن را int برمیگرداند.

```
length(arr)
```

Min: آرایه را ورودی میگیرد و در خروجی کوچکترین عنصر آن را برمیگرداند .

```
min(arr)
```

Max: آرایه را در ورودی می گیرد و در خروجی کوچکترین عنصر ارایه را برمیگرداند .

```
max(arr)
```

Index: تابع index برای دسترسی به یک عنصر مشخص در یک آرایه استفاده می‌شود. این تابع دو ورودی دریافت می کند.

```
index(arr, i)
print(Index(nums, 5))
```

در صورت وجود نداشتن اندیس با ارور مواجه شویم.

Multiply: دو آرایه نظیر به نظیر ضرب می شوند.

```
multiply(arr1, arr2)
```

در صورت مساوی نبودن سایز دو آرایه با ارور مواجه شویم.

Add: دو آرایه نظیر به نظیر جمع می شوند.

```
add(arr1, arr2)
```

Subtract: دو آرایه نظیر به نظیر تفریق می شوند.

```
subtract(arr1, arr2)
```

Divide: دو آرایه نظیر به نظیر تقسیم می شوند.

```
divide(arr1, arr2)
```

- **Array Calculations**:

محاسبات آرایه‌ای به ما این امکان را می‌دهد روی کل یک آرایه، عملیات ریاضی را اعمال کنیم. این عملیات می‌تواند شامل موارد زیر باشد:

1. جمع یک عدد با تمامی عناصر یک آرایه (array + number)
2. تفریق یک عدد از تمامی عناصر یک آرایه (array - number)
3. ضرب تمامی عناصر یک آرایه در یک مقدار عددی (array * number)
4. تقسیم تمامی عناصر یک آرایه بر یک مقدار عددی (array / number)

```
array nums = [10, 20, 30];
array result = nums - 5;
print(result); // [5, 15, 25];
```

- **Try / Catch**

از try / catch دراین زبان برای مدریت خطاهای احتمالی استفاده میشود.

```
try {
    int x = 1 / 0;
} catch (Error e) {
    print("Error: Division by zero!");
}
```

- **نوشتار تک خطی**

این سینتکس امکان تعریف ساده تر شرط را فراهم میکند.

```
x = y if z > 5 else w;
```

که معادل :

```
if (z > 5) {
   x = y;
else {
   x = w;
}
```

- **Pattern Matching**:

این روش مشابه switch-case در زبان‌های دیگر است.

```
match x {
   0 -> print("zero"),
   1 -> print("one"),
   _ -> print("other")
}
```

- **Array Iteration** :

پیمایش آرایه به ما این امکان را می‌دهد که به تمام عناصر یک آرایه دسترسی داشته باشیم و روی هر عنصر عملیاتی انجام دهیم. این کار معمولاً با استفاده از حلقه‌های for و foreach انجام می‌شود.

```
for (int i=0; i<10; i++){
    a[i]++;
}
```

نکات مهم

- از زبان C یا C++ برای توسعه کدها استفاده کنید.
- توسعه هر بخش در فایل‌های جداگانه انجام شود که قابلیت ارزیابی مجزا را داشته باشد.
- کدها خوانایی مناسبی داشته باشند و پیشنهاد می‌شود به درستی کامنت‌گذاری شود.
- پروژه به صورت تیمی قابل انجام است. اندازه تیم‌ها بین ۲ الی ۳ نفر قابل قبول است و در فاز بعدی، امکان تغییر اعضای تیم وجود ندارد.
- همه اعضای تیم باید در انجام پروژه مشارکت داشته باشند و تسلط هر فرد جداگانه ارزیابی خواهد شد.
- جهت پیاده‌سازی درست و کامل پروژه، پیشنهاد می‌شود اسناد مرتبط با سایت با دقت مطالعه شده و همچنین سه فصل اول کتاب learn llvm 12 مطالعه شود. کدهای نمونه و روش ارائه مطالب در کتاب به درک شما از پیاده‌سازی کامپایلر مینی‌مال کمک خواهدکرد.
- بخش قابل توجهی از نمره پروژه، مربوط به اجرای صحیح تست‌ها است که در زمان ارائه به شما داده خواهد شد. صحت عملکرد هر بخش از پروژه نیز جداگانه بررسی می‌شود و در صورتی که در بخشی از پروژه اشکال داشته باشید، نمره‌ی باقی قسمت‌ها را دریافت می‌کنید.

موفق باشید