ASD

Proqramlaşdırma dili

Versiya 3.0

ASD Proqramlaşdırma Dili

Müasir dövrdə məsələlərin kompüterlərlə həlli geniş yayılıb. Belə ki, kompüterlər insanların bacara bilməyəcəyi sürətdə hesablamalar apardıqları kimi məsələləri də çox tez həll edə bilərlər. Lakin kompüterlər insan kimi düşünə bilmir və məsələnin həlli üçün komanda verilməsinə ehtiyac duyur. Onları müxtəlif vəziyətlərə qarşı proqramlaşdırmalı biri olmalıdır. Bilindiyi kimi kompüterlər ikilik say sistemində işləyir və məhz əmrləri də ikilik say sistemi şəklində ötürülən maşın kodundan alırlar. Maşın kodunu insanların başa düşəcəyi şəkildə yazmaq üçün proqramlaşdırma dilləri yaradılmışdır. İlk dillərin başa düşülməsi çətin olsa da, sonralar insanların asanlıqla anlayacağı dillər yazılmışdır. Bununla belə, proqramlaşdırmaya ilk başlayanlar bəzi məqamlarda çətinlik çəkə bilərlər: dilin sintaksisi çətin ola bilər, yazılan proqramların universal dildə - ingiliscə olması bu dili bilməyənlər üçün qarışıq gələ bilər.
Məhz ASD proqramlaşdırma dili bu məqsədlə yaradılmışdır. Dil azərbaycanlıların alqoritmləri və proqramları rahat şəkildə anlaması üçün yazılmışdır. Dilin sintaksisi rahat və anlaşılandır. ASD sadəcə 10(11) açar söz və 22 funksiyadan ibarətdir. Əvvəlki versiyalarda ASD struktursuz dil olsa da, son versiya ilə dil strukturlu dilə çevrilmişdir.

ASD dilinin quraşdırılması

ASD dili müxtəlif əməliyyat sistemlərində işləyə bilir. Bunun üçün müxtəlif yollar yaradılmışdır.

ASDroid

Androiddə ASD kodlarını yazmaq üçün istifadə olunur. Qısaca oflayn olaraq kod yazmaq üçün vebsaytın aplikasiya şəklinə gətirilmiş formasıdır. Lakin ASDroid proqramında fayl endirmək və ya yükləmək mümkün deyildir.

Vebsayt

Siz ASD dilində yazılmış proqramı dilin vebsaytından çalışdıra bilərsiniz. Bunu üçün axtarış çubuğuna “asddili.glitch.me/v3.html” yazmağınız kifayətdir. Əvvəldə qeyd etdiyimiz kimi bu vebsayt dilin 3.0 versiyasında olan kodların yazılması və çalışdırılması ilə bağlıdır. Vebsaytda həmçinin faylları yükləyə və yaxud endirə bilərsiniz. Vebsayt bütün əməliyyat sistemlərini dəstəkləyir. Yazdığınız kodların daha anlaşılan olması üçün sintaksis rəngləndirməsi mövcuddur.

Terminal

Həmçinin ASD kodları Linux terminalında da çalışdırıla bilər. Lakin unutmayın ki, digər üsullardan fərqli olaraq ASD burada sizə proqram yazıçısı ilə təmin etməyəcək. ASD dilini terminala quraşdırmaq üçün bir neçə komanda yazmaq lazımdır. (Qeyd: əvvəlcədən **git** və **python** paketlərini yükləyin.)

```
$ git clone https://github.com/Fma-coder/asdlang
$ cd asdlang
$ . asdlang.sh
```

Artıq ASD istifadəyə hazırdır. Lakin hər dəfə bunu etmək lazım gəlməyəcək. ASD dilini istifadə edəcəyiniz zaman son iki sətri yazmağınız kifayətdir. Yazdığımız Linux kodları əməliyatın sonunda bizə **asd** funksiyasının istifadəsinə icazə verir.

```
$ asd <fayl>
```

Bu zaman `<fayl>` hissəsinə çalışdırmaq istədiyiniz faylın yerini yazın. ASD dili lazımi kodları ilə yanaşı özü ilə ASD dilində yazılmış “Qabarcıqlı nizamlama” proqramını da gətirir. Bu kodu çalışdıraq:

```
$ asd bubblesort.asd
```

Bu proqramın necə işlədiyinə sonrakı bölmələrdə baxacayıq. Terminalda ASD dilində fayl yazmaq üçün bu komandaları daxil etməyiniz kifayətdir:

```
$ touch <faylın adı>.asd
$ nano <faylın adı>.asd
```

“yaz” açar sözü

“yaz” açar sözü ASD dilində iki xaricetmə üsullarından biridir. Bu açar sözün vasitəsilə xaric edilmiş arqumentlər proqram sonuna qədər ekranda qalır. Açar sözün quruluşu çox sadədir:

```
yaz arqument
```

Gəlin bu açar sözün istifadəsi ilə ilk proqramımızı yazaq. Yazacağımız proqram standart olaraq “Salam Dünya” proqramı olacaq:

```
yaz “Salam Dünya!”
```

Proqramı çalışdırsaq, ekranda “Salam Dünya!” sözünün yazıldığını görərik. Əgər proqramı genişləndirmək istəsəniz, genişləndirin:

```
yaz “Salam Dünya!#y#”
yaz “Salam ASD!”
```

Bu zaman isə siz alt-alta yazılmış iki müvafiq cümləni görəcəksiniz. Proqram sətirindən göründüyü kimi, birinci sətirdə yazılan amma proqram çıxışında yazılmayan bir simvol var - “#y#”. Bu simvol yeni sətrə keçilməsini təmin edir, əgər o olmasa idi, biz bitişik iki cümlə görərdik. Bu proqramı tək sətrə də gətirə bilərik:

```
yaz “Salam Dünya!#y#Salam ASD!”
```

Proqramda “yaz” sözü dəyişməz olaraq qalır, lakin arqument istənilən tipdə - simvol, rəqəm və kəsr tipində ola bilər. Bu tiplərdən istifadə edərkən bəzi məqamlarda diqqətli olmaq lazımdır. Belə ki, Siz “+” işarəsini simvol tipli arqumentlərin arasında işlətsəniz, onlar bitişəcək. Lakin bu operator rəqəm, yaxud kəsr tipli arqumentlər arasında işlədilsə, verilən arqumentlər toplanacaq. “+” operatoru simvol-rəqəm və simvol-kəsr arasında işlənə bilməz və işlədilərsə proqramın işlədilməsində problem yaranar.

Digər məsələ isə dırnaqların istifadədir. Dırnaqlar arqumenti simvol tipinə çevirir və onun işlənməsinin qarşısını alır. Dırnaq içində yazılmayan arqumentlər dəyişənlər, rəqəmlər və kəsrlərdir. Proqram nümunəsində dırnaq içindəki arqumentlərə toxunulmur, xaricdəkilər isə işlənilir:

```
yaz “2+5=”+simvol(2+5)
```

(Çıxış: 2+5=7)

Diqqət: 7 rəqəm tipində olduğu üçün onu simvola çevirib bitişdirdik.

“dəyişən” açar sözü

Təsəvvür edin, elə proqram yazırsınız ki, böyük ədədlər üzərində əməllər aparır, və yaxud sizə təsadüfi seçilmiş hərflər yığını verilir. Təbii ki, bu verilənləri yadda saxlamaq çətin olacaq. Bunun üçün ASD dilində dəyişənləri istifadə edə bilərsiniz. Dəyişən elan etmək üçün çox sadə sintaksisə əməl etməyiniz lazımdır.

```
dəyişən dəyişənin adı = dəyişənin qiyməti
```

“dəyişən” açar sözü dəyişilməz olaraq qalır. Dəyişənin adı və qiyməti isə proqramçı tərəfindən müəyyən edilir. Bununla belə, dəyişən adı rəqəmlə başlaya bilməz və dəyişənin adı sadəcə hərflər və rəqəmlərdən ibarət ola bilər. Məsələn, dəyişən istifadə edilmiş bir proqrama baxaq:

```
dəyişən a = 5
dəyişən b = 4
yaz a*b
```

Yazdığımız proqramın cavabı isə 20 olacaq. Proqramda istifadə etməyimizə baxmayaraq, sizə tövsiyəmiz təkhərfli və ya məna daşımayan sözləri dəyişən adı olaraq istifadə etməməyinizdir. Bununla siz kodunuzun daha anlaşıqlı və mənalı olmasını təmin edə bilərsiniz. Digər bir tərəfdən bütün hesab əməllərini “yaz” açar sözündə aparmağa ehtiyac yoxdur. Bunu dəyişənin qiyməti hissəsində də aparmaq olar. Məsələn yazdığımız proqramda “c” adlı yeni dəyişən yaradıb ona **a*b** qiymətini mənimsədə bilərik. Sonra isə “yaz” açar sözü ilə **c** dəyişənin qiymətini ekrana çıxara bilərsiniz. Gəlin, başqa bir proqrama baxaq:

```
dəyişən xitab = “ASD”
yaz “Salam ” + xitab
```

Cavabı isə “Salam Dünya” olacaq(Cavabı gözləməyin, sınayın).

“mesaj” açar sözü

Bu açar söz ASD dilinin terminal versiyasında mövcud deyil. Vebsayt və ASDroid versiyasından terminala keçid edərkən bu açar sözü “yaz” ilə əvəz edin. Gələcək versiyalarda açar sözün dildən silinməsi nəzərdə tutulur. İndilik isə “mesaj” açar sözünün tək məqsədi “sorğu” açar sözünün istifadəsi zamanı mesajların çatdırılmasıdır. Lakin siz bu

açar sözü yazılan sözü dərhal silmək üçün də istifadə istifadə edə bilərsiniz.

```
mesaj arqument
```

Bu açar söz ilə bağlı çox bəhs etməyəcəyik. Çünki arqument “yaz” açar sözü ilə eyni xüsusiyyətləri daşıyır.

“sorğu” açar sözü

Fikirləşin ki, kvadrat hesablayan proqram yazırsınız. Bu proqram, yəqin ki, belə olacaq:

```
dəyişən ədəd = 7
yaz ədəd*ədəd
```

Sonra isə siz müxtəlif ədədlərin kvadratlarını hesablamaq istəyirsiniz. Lakin hər dəfəsinə proqram kodunu dəyişdirmək məcburiyyətində qalırsınız. Bu, həm yorucu, həm də istifadəçinin edə bilməyəcəyi bir iş olar. Məsələnin həllində köməyinizə “sorğu” açar sözü gəlir. Bu açar söz istifadəçidən müəyyən edilmiş sualı soruşur və cavabını dəyişənə ötürür.

```
sorğu dəyişən = sual
```

Bu açar sözü əvvəlki proqramımıza tətbiq edərək onu dəyişdirək:

```
sorğu ədəd = “Ədədi daxil edin.”
dəyişən ədəd = rəqəm(ədəd)
yaz ədəd*ədəd
```

İndi isə sətirbəsətir açıqlayaq. Birinci sətir istifadəçidən sualı soruşur və “ədəd” dəyişəninə cavabı mənimsədir. Mənimsədilən qiymət simvol tipində olduğu üçün ikinci sətirdə onu “rəqəm()” funksiyası ilə rəqəm tipinə çeviririk. Son sətirdə isə ədədin kvadratını hesablayıb ekrana yazdırırıq.

“əgər” açar sözü

Deyək ki, siz bankomat üçün proqram yazırsınız. Bunun üçün siz istifadəçiyə müəyyən seçimlər verib verilən cavablara uyğun olaraq əməliyyatları icra etməlisiniz. Proqram ilk olaraq asan görünə bilər, amma seçimləri yoxlamaq üçün yeni açar sözə ehtiyac duyursunuz. Bu işi ASD dilində “əgər” açar sözü yerinə yetirir. “əgər” açar sözünün quruluşu belədir:

```
əgər şərt:əmr
```

Digər açar sözlərdə olduğu kimi, burada da “əgər” açar sözü sabitdir. Şərti və əmri isə proqramçı təyin edir. Şərt bölməsində istifadə oluna biləcək müqayisə işarələri aşağıdakı cədvəldə göstərilmişdir.

==	Bərabərdir	<	Kiçikdir
!=	Bərabər deyil	>=	Böyük bərabərdir
>	Böyükdür	<=	Kiçik bərabərdir

Bununla siz, sadə bilik yarışı proqramı yaza bilərsiniz. Məsələn:

```
sorğu cavab = “AXC neçənci ildə müstəqillik qazanmışdır?#y#1 - 1922#y#2 - 1991#y#3 - 1918#y#4 - 1917”
əgər cavab == “3”:mesaj “Doğrudur.”
```

Bəli, biz cavabı yoxlamağı bacardıq, bəs cavab düzgün olmayanda nə edəcəyik? ASD dilində bunun üçün “əgər-yox” zənciri istifadə olunur.

```
əgər şərt1:əmr1
yox
əgər şərt2:əmr2
yox
əgər şərt3:əmr3
```

Bu zəncirdə əgər şərt1 doğru olarsa əmr1 yerinə yetiriləcək, yox olmazsa onda zəncir ikinci əgərə keçəcək. şərt2 doğru olarsa əmr2 yerinə yetirilir, olmazsa üçüncü əgərə keçirilir. Lakin bu demək deyil ki, siz zəncirdə yalnız üç əgər istifadə edə bilərsiniz. Əgər-yox zənciri istənilən qədər şərt qəbul edir və asanlıqla genişlənir.

Bəzən isə bir neçə şərt və yaxud əmr vermək lazım gəlir. ASD dilində birdən çox şərt yazmaq üçün müxtəlif məntiqlər var. Məsələn && “və” məntiqini göstərir. Bu o deməkdir ki, əmrin yerinə yetirilməsi üçün verilən şərtlərin hər ikisi(və ya daha çoxu) doğru olmalıdır. “||” simvolu isə “və ya” məntiqini göstərir, yəni verilən şərtlərdən ən azı birinin doğru olması əmrin yerinə yetirilməsi ilə nəticələnir. Nümunələrə baxaq:

```
əgər (4==4) && (10<5):yaz 1
əgər (4==4) || (10<5):yaz 2
```

Verilən proqramın nəticəsində ekrana 2 rəqəmi yazılacaq, çünki birinci sətirdə 1 rəqəminin yazılması üçün bütün şərtlər doğru olmalı idi, amma

ikinci şərt yanlışdır. Lakin eyni şərtlər ikinci sətirə yazıldıqda ekrana 2 yazılır. Bunun səbəbi isə "və ya" məntiqindən istifadədir - beləliklə şərtlərdən biri doğru olduğundan ekranda 2 rəqəmi göstərildi.
Şərtlərdən başqa bəzən əmrlərin də çoxalması ehtiyacı yaranır. Bunun üçün çoxsətirli əgər və yaxud funksiyalar(digər adlandırması altproqram, gələcək mövzularda bəhs ediləcək) istifadə edilə bilər. Çoxsətirli əgər daha geniş istifadə olunur, lakin onun bir çatışmazlığı var - çoxsətirli əgərin əmrləri içində yenidən əgər istifadə edilə bilməz. Quruluşu əgərlə eynidir:

```
əgər şərt:əmr1;əmr2;əmr3;....
```

Çoxsətirli əgərin köməyi ilə bir proqram yazaq:

```
dəyişən eded = 4
əgər eded>3:dəyişən eded=eded*eded;yaz eded
```

Proqramda eded dəyişəni üçdən böyük olduğu üçün yenidən həmin dəyişənə onun kvadratı mənimsədilir və cavab ekrana yazılır.

"funksiya" və "çağır" açar sözü

Çox zaman proqram yazarkən bir kod fraqmenti dəfələrlə istifadə edilir. Bu fraqmentin ölçüsü böyük, və ya kiçik ola bilər. Ölçü böyük olarsa, onu dəfələrlə yazmaq proqramçını həm həvəsdən salar, həm də onun məhsuldarlığı aşağı düşər. Bu problemin həlli üçün ASD dilində "funksiya" açar sözü var. Quruluşu belədir:

```
funksiya adı[
…
…
]adı
```

Gəlin, yeni bir funksiya yazaq. Bu funksiya ekrana üçbucaq çıxarır:

```
funksiya üçbucaq[
yaz "*#y#**#y#***"
]üçbucaq
```

Funksiyanı çağırmaq üçün "`çağır üçbucaq`" yazmağımız kifayətdir. Lakin digər dillərdə olduğu kimi ASD dilində arqumentli və döndürmə(return) tipli funksiyalar yoxdur. Arqumentli funksiya yaratmaq üçün dəyişənlərdən istifadə olunur:

```
funksiya kvdrt[
yaz a*a
]kvdrt
```

Bu fraqmentdə biz funksiyanı elan etdik. Funksiyada **a** dəyişəni yazılmasına baxmayaraq o elan edilməyib. Bunun üçün narahat olmağa səbəb yoxdur, çünki çağırılana qədər funksiyalar sadəcə elan edilir, işlədilmir.

```
dəyişən a = 4
çağır kvdrt
```

Proqramın cavabında ekranda 16 rəqəmi yazılacaq. Çünki biz funksiya çağırılmamışdan əvvəl **a** arqumentinə qiymət mənimsətdik və funksiya bu qiyməti işləyib cavabı ekrana yazdırdı.

Qeyd: Funksiya adlarını yenidən dəyişən kimi istifadə etmək olmaz, funksiyalar dəyişənlərdə saxlanılır.

“dövr” açar sözü

Fikirləşin ki, siz istifadəçi tərəfindən daxil edilmiş ədədə qədərki ədədləri ekrana yazdırmaq istəyirsiniz. Təbii ki, bu ədəd 1000 də olar bilər, 6 da. Bu o deməkdir ki, proqramın ekrana nə qədər ədəd çıxaracağı naməlumdur, bunu yalnız proqramın işlədiyi zaman istifadəçi və proqram bilə bilər. Proqramın kodu nə qədər təkrar edəcəyini ona bildirmək üçün ASD dilində “dövr” açar sözü mövcuddur:

```
dövr ad=>(şərt){
..
..
..
}ad
```

Açar söz şərt doğru olduğu bütün hallarda kodu yenidən işlədəcək. Ad hissəsi proqramçıdan asılıdır, amma istifadə edilən dəyişənin istifadə edilməsi daha məqsədəuyğundur. İndi isə proqramımızın kodunu yazaq.

```
sorğu ədəd = “Ədədi daxil edin.”
dəyişən ədəd = rəqəm(ədəd)
dəyişən d = 1
dövr ədəd=>(ədəd >= d){
yaz d
yaz “#y#”
dəyişən d = d + 1
}ədəd
```

Birinci sətir ədədi istifadəçidən alır və ikinci sətir onu rəqəmə çevirir. **d** dəyişəni sayğac rolunu oynayır. Dövr şərti yoxlayır və **d** dəyişənini ekrana yazır(yeni sətir əlavə edərək), sonra isə **d** dəyişənini bir vahid artırır ki, sonsuz dövr yaranmasın.
Bu açar sözdən istifadə edərkən bəzi məqamlarda ehtiyatlı olmaq lazımdır. Birdən çox dövr eyni cür adlandırıla bilməz. Şərtdən asılı olmayaraq dövr ən azı bir dəfə yerinə yetirilir və sayğacı düzgün yazdığınızdan əmin olun ki, sonsuz dövr yaranmasın.

"get" və "dayan" açar sözləri

Bəzən proqramlarda istifadəçidən gələn yanlış daxiletmələr qarşısında proqramı dayandırmaq lazım gəlir. Bu zaman "dayan" açar sözünün istifadəsi məqsədə uyğundur. Bu açar söz heç bir arqument qəbul etmir və həmin sətirdə proqramı dayandırır:

```
dayan
```

Sonuncu açar söz olan "get" açar sözü arqument olaraq sətrin nömrəsini qəbul edir. Lakin arqument həmişə rəqəm olmaq məcburiyyətində deyil, o, dəyişəndə saxlanılmış rəqəm də ola bilər.

```
get sətir
```

ASD dilinin əvvəlki versiyalarında geniş istifadə olunmasına baxmayaraq, sonuncu versiyada "funksiya" və "dövr" açar sözlərinin dilə gətirilməsi ilə istifadəsi azalıb. Məntiqi cəhətdən faydalı olsa da, yeni sətir əlavə edərkən sətir nömrəsinin dəyişməsi ilə bağlı problemlər proqramçıya çətinliklər törədə bilər.

Daxili funksiyalar

ASD dilində olan daxili funksiyalar sadəcə dəyəri qaytarır. Bunu nəzərə alaraq onları istifadə edərkən qiymətini yazdırmağı və yaxud dəyişənə mənimsətmək daha yaxşı olar.

1. rəqəm(a) - Verilən arqumenti rəqəm tipinə çevirir.
2. kəsr(a) - Verilən arqumenti kəsr tipinə çevirir.
3. simvol(a) - Verilən arqumenti simvol tipinə çevirir.
4. modul(a) - Verilən arqumentin modulunu hesablayır.
5. random(a, b) - a ilə b arasında təsadüfi ədəd seçir.
6. yerləşdir(a, b, c) - a-da olan b-ləri c ilə əvəz edir.

7. kiçik(a) - Verilən arqumentin hərflərini kiçildir.
8. böyük(a) - Verilən arqumentin hərflərini böyüdür.
9. yuvarlaq(a) - Verilən ədədi yuvarlaqlaşdırır.
10. kök(a) - Verilən ədədin kökünü hesablayır.
11. sil(a, b) - a-da olan b-ləri silir.
12. kəs(a, b) - a sətrini b qədər kəsir.
13. ayır(a, b) - a sətrində olan b-ləri silir və sıra yaradır.
14. daxildir(a, b) - b-nin a-ya daxil olub-olmadığını yoxlayır.
15. say(a, b) - a-da olan b-lərin sayını tapır.
16. çevir(a, b) - a rəqəmini b-lik say sisteminə çevirir.
17. uzunluq(a) - Arqumentin uzunluğunu tapır.
18. əlavət(a, b) - a sırasına b elementini əlavə edir.
19. düz(a) - a sırasını əlifbaya görə düzür.
20. ədəddüz(a) - a sırasını kiçikdən böyüyə düzür.
21. birləş(a, b) - a sırasının bölünmələrini b ilə birləşdirir.
22. indeks(a, b) - b-nin a-dakı sırasını tapır.

Sıralar

Sıralar xüsusi dəyişən tipidir. Sıralar vasitəsilə bir dəyişənə birdən çox qiymət mənimsədilə bilər. Qiymətlər sıra şəklində saxlanılır və onları istifadə etmək üçün onun sıradakı nömrəsini bilmək lazımdır.

```
dəyişən qafqaz = [“Azərbaycan”, “Gürcüstan”, “Ermənistan”]
yaz qafqaz[0]
```

Proqramın cavabı Azərbaycan olacaq. Çünki sıralarda elementlər sıfırdan başlayaraq mənimsədilir. Beləliklə Ermənistan sözünü yazmaq üçün biz proqrama **qafqaz[2]** əlavə etməliyik. Sıraların elementlərini biz dəyişən və sorğu açar sözü ilə dəyişdirə bilərik:

```
dəyişən qafqaz[0] = “Odlar yurdu”
yaz qafqaz
```

Ekrana isə Odlar yurdu, Gürcüstan, Ermənistan yazılacaq. Siz həmçinin sıra nömrələrini dəyişənlərlə ifadə edə bilərsiniz.

```
dəyişən nömrə = 1
dəyişən qafqaz[nömrə] = “Sakartvelo”
yaz qafqaz
```

Şərhlər

Şərhlər ASD-də kodu anlamağa və izah etməyə kömək edir. Proqram işlədilərkən şərh sətirləri proqram tərəfindən nəzərə alınmır. Şərh sətir nöqtə ilə başlayır. Məsələn,

```
.a dəyişəninə 5 qiymətini mənimsədirik.
dəyişən a = 5
```

Proqram nümunəsi - Qabarcıqlı nizamlama

```
funksiya bubblesort[
dəyişən k=0
dövr k=>(k<uzunluq(sirasi)){
dəyişən j=0
dövr j=>(j<(uzunluq(sirasi)-k-1)){
əgər sirasi[j]>sirasi[j+1]:dəyişən müvəq=sirasi[j];dəyişən
sirasi[j]=sirasi[j+1];dəyişən sirasi[j+1]=müvəq
dəyişən j=j+1
}j
dəyişən k=k+1
}k
yaz sirasi
]bubblesort
dəyişən sirasi=[9, 4, 2, 3, 8, 7, 10, 1, 6, 5]
çağır bubblesort
.Proqramın cavabı: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```