315 Programlama Dilleri

Yrd. Doç. Dr. Ahmet Arif AYDIN

Özet

- Aritmetik ifadeler
- Çok amaçlı (overloaded) operatörler
- Tip Dönüşümleri
- İlişkisel ve Mantıksal ifadeler
- Kısa Devre değerlendirme

Unary Operator

Bir terim üzerinde işlem gerçekleştirir.

```
public static void main(String[] args){

        int result = +1; // result is now 1
        System.out.println(result);
        result--;  // result is now 0
        System.out.println(result);
        result++; // result is now 1
        System.out.println(result);
        result = -result; // result is now -1
        System.out.println(result);
        boolean success = false;
        System.out.println(success); // false
        System.out.println(!success); // true
    }
```

Unary Operator

*test2.java

```java
public class test2 {

    public static void main(String[] args){
        int sonuc = +1; // sonuc'a +1 degerini atadi
        System.out.println(sonuc);
        sonuc+=1; //  sonuc'un  değeri 1 artttırıldı
        System.out.println(sonuc);
        sonuc--;
        System.out.println(sonuc);
        sonuc++;
        System.out.println(sonuc);
        sonuc = -sonuc;
        System.out.println(sonuc);
        boolean success = false;
        System.out.println(success);
        System.out.println(!success);
    }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> test2 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_1C
1
2
1
2
-2
false
true
```

Bir terim üzerinde işlem gerçekleştirir.

Binary Operator

İki terim üzerinde

işlem gerçekleştirir

Equal (==)
Not equal (!=)
Less than (<)
Greater than (>)
Greater than or equal to (>=)
Less than or equal to (<=)
Logical AND (&&)
Logical OR (||)
Plus (+)
Minus (-)
Multiplication (*)
Divide (/)

Ternary Operator

3 tane terim üzerinde işlem gerçekleştirir.

```
#include <stdio.h>

main()
{ int a , b, c;
a = 10;
b = 5;
printf( " Deger %d\n", (a == 1) ? 20: 30 );
printf( " Deger %d\n", (b == 10) ? 20: 30);
}
```

```
JAVA
deger= (a < b) ? a : b;
```

Statement-Level Control Structures

(Komut seviyeli kontrol yapıları)

Giriş

- Imperative dillerdeki işlemler ifadelerin değerlendirilmesi ve sonuçların değişkenlere atanmasıyla gercekleştirilir.
- Programlama dillerinde bulunan bazı yapılar yardımıyla hesaplamalar ve aritmetik işlemler daha güçlü ve esnek hale getirilir.

Giriş

Alternatif yollar arasında seçim yapılarak akış kontrolu belirlenebilir

Veya

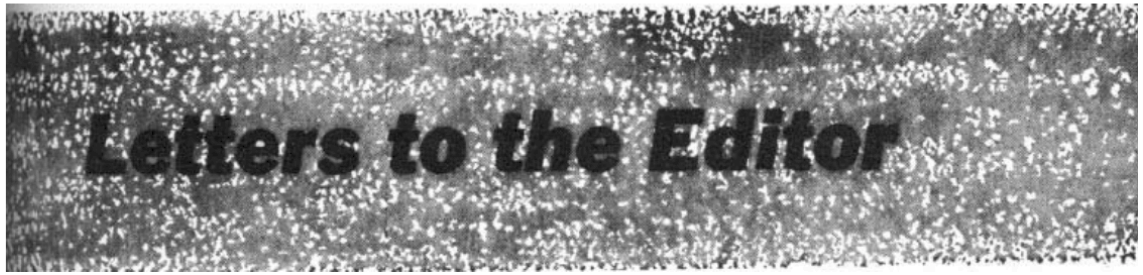
Belirlenen ifade dizilerinin tekrarlı olarak işlenmesi saglanabilir

control statements
(kontrol ifadeleri)

Control Statements

- Ilk defa Fortran dilinde kullanılmıştır. (1950)
- 1960-1970 yılları arasında control ifadeleri uzerinde akademik calışmalar yapılmıştır.
  - Goto statement consıdered harmful
  - Selectable Goto ifadesi

SONUÇ: Sartsız olarak kullanılan kontrol ifadelerinin gereklı olmadığı



Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing
CR Categories: 4.22, 5.23, 5.24

EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the

dynamic progress is only cl
call of the procedure we r
we can characterize the pr
textual indices, the lengtl
dynamic depth of procedu

Let us now consider repe
or **repeat** *A* **until** *B*). Log
superfluous, because we c
recursive procedures. For

Control Statements

Programcıların kontrol yapılarından genel beklentileri

- Readability (okunabilirlik)
- Writeability (yazılabilirlik)

Selection Statements

Seçme Komutları

Program içerisinde bir veya daha fazla çalışma yolu (execution path) izlenmesine olanak sağlar.

İki ana kategori altında incelenir:

1. Two-way , n-way
2. Multiple selection

Selection Statements

Two-way , n-way

```
if kontrol ifadesi then
      ifade1
else
     ifade2
```

C, Python, ve C++ dillerinde kontrol ifadesi boolean veya sayısal olabilir

Ada, Java, Ruby, C# da Boolean olmak zorunda!

Selection Statements

Two-way , n-way

RUBY

```
a = 10 * rand(2)

if a < 5
      puts "#{a} -- 5 den küçük"

elsif a > 7
      puts "#{a} --  7 den büyük"

else
      puts ”Programlama Dilleri "

end
```

RUBY

```
unless a > 5
      puts "a is less than or equal to 5"
else
      puts "a is greater than 5"
end
```

Selection Statements

PYTHON

```
if x > y :
        x= y
        print  " atama "
```

```
if x==100 :
                işlemler 1
elif x==200:
                işlemler 2
elif x==300:
                işlemler 3
else:
                işlemler 4
```

Selection Statements

```
JAVA
If (mantıksal ifade1) {
// mantıksal ifade1 ın sonucu 1 ise çalışır
} else if(mantıksal ifade1 2) {
 // işlemler
} else if(mantıksal ifade1 3) {
// işlemler
} else {
// yukarıdaki şartlar dogru degılse calışır
}
```

Selection Statements

İç içe seçim yapısı

```
if (toplam== 0)
    if (sayac== 0)
        sonuc= 0;
else
    sonuc= 1;
```

else ifadesi hangi if e ait ?

Selection Statements

İç içe seçim yapısı

```
if (toplam== 0)
    if (sayac== 0)
        sonuc= 0;
else
    sonuc= 1;
```

JAVA

```
if (sum == 0) {
     if (count == 0)
     result = 0;
}
else {
     result = 1;
}
```

C , C++ , C#

```
if (sum == 0) {
     if (count == 0) {
          result = 0;
     }
else {
      result = 1;
     }
}
```

RUBY

```
if sum == 0 then
     if count == 0 then
          result = 0
     end
else
     result = 1
end
```

PYTHON

```
if sum == 0 :
     if count == 0 :
          result = 0
else:
     result = 1
```

Selection Statements

Multiple Selection

Birden fazla seçeneğin seçilip işlem yapılmasını sağlar.

Selection Statements

Multiple Selection

```
switch (ifade) {

    case sabit-ifade 1:
        işlemler 1;
    ...

    case sabit-ifade n:
         işlemler n;
        break;
    [default:
        işlemler n + 1]
}
```

Selection Statements

Multiple Selection

```c
#include <stdio.h>

int main () {

   /* local variable definition */
   char ders_notu = 'F';

   switch(ders_notu) {
      case 'A' :
         printf("Mükemmel!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("iyi iş\n" );
         break;
      case 'D' :
         printf("Geçtin\n" );
         break;
      case 'F' :
         printf("Seneye görüşürüz! \n" );
         break;
      default :
         printf("Yanlıs not girdiniz\n" );
   }

   printf("Notunuz  %c\n", ders_notu );

   return 0;
}
```

```
Seneye görüşürüz!
Notunuz  F
```

https://www.tutorialspoint.com/cprogramming/switch_statement_in_c.htm

Selection Statements

Multiple Selection

```
RUBY
case
    when şart-ifadesi then işlem 1
    ...
    when şart-ifadesi then işlem 2

    [else işlem 3]
end
```

```
PYTHON
if x==100 :
        ifade
elsif x==200:
        statement(s)
elsif x==300:
        statement(s)
```

İterative Statements

Tekrarlayan Komutlar

İfade veya ifadeler topluluğunu 0 veya n defa çalıştırılmasını sağlayan yapılara ***iterative statements*** denir.

Iterative yapılar döngü (loop) olarak bilinir.

Programlama da cok onemli yapılardır.

Döngünün olmadığını varsayarsak yapılacak olan işlemin satır satır yazılması gerekecekti..

- Programın yazımı cok zaman alacaktı
- Faydalı program parcacıkları cok fazla yer ve hafıza alanını kullanmış olacaktı.

İterative Statements

Tekrarlayan Komutlar (Sayac kontrollü döngüler)

Döngü değişkeni bulunur

Degişkenin ılk degeri , son değeri ve değişkenin arttırım değeri bulunmaktadır.

C , C++

```
for (değişken= 1; değişken <= 10; degişken++) {
    ...
}
```

Sonsuz döngü
Break
Döngü değişkeni değeri belirlenir

İterative Statements

Tekrarlayan Komutlar (Sayac kontrollü döngüler)

Döngü değişkeni bulunur

Degişkenin ılk degeri , son değeri ve değişkenin arttırım değeri bulunmaktadır.

```
for (değişken= 1; değişken <= 10; degişken++) {
    ...
}
```

Kontrol ifadesi JAVA , C# da boolean olmak zorunda
C++ da boolean veya sayisal deger olabilir
C de boolean deger olamaz

İterative Statements

Tekrarlayan Komutlar (Sayac kontrollü döngüler)

meyveler= **['elma'**, "portakal", "armut, "uzum"**]**

PYTHON

```
for sayı in [2, 4, 6]:
    print sayı
```

RUBY

```
for k in 0..5
    puts " k nın degeri #{k}"
end
```

```
meyveler.each do |meyve|
    puts "meyve: #{meyve}"
end
```

İterative Statements

Tasarım problemleri

önce döngü degişkeni mi arttırılacak yoksa karşılaştırma işlemimi gerçekleşecek?

Döngü değişkeninin kapsamı ?

```
for (değişken= 1; değişken <= 10; degişken++) {
    ...
}
```

İterative Statements

Tasarım problemleri

önce döngü degişkeni mi arttırılacak yoksa

karşılaştırma işlemimi gerçekleşecek?

```
while( x < 20 )
{
 printf("deger : %d\n", x);
 x++;
}
```

```
int deger =7
do {
      System.out.print(deger);
}
while( deger < 5);
```

C , C++ , JAVA

İterative Statements

Break , Continue

```c
#include <stdio.h>

int main() {

    int a;
    for (a=0; a<=10;a++){
        printf("merhaba");
        continue;
    }

    return 0;
}
```

```
merhabamerhabamerhabamerhabamerhabamerhabamerhaba
```

İterative Statements

Break , Continue

```c
int main() {

    int a;
    for (a=0; a<=10;a++){
        printf("merhaba");
        break;
    }

    return 0;
}
```

```
merhaba
```

İterative Statements

Break , Continue

```c
#include <stdio.h>

int main()
{

    int a ;
    for ( a=0 ; a<=50 ; a++) {
        if (a % 2)
        printf("%d\n" , a);
    }
}

```

```
0
1
2
..
50
```

İterative Statements

Break , Continue

```c
#include <stdio.h>

int main()
{

    int a ;
    for ( a=0 ; a<=50 ; a++) {
        if (a % 2) continue;
        printf("%d\n" , a);
    }
}

```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

Continue döngünün sonuna gider fakat döngüden çıkmaz

İterative Statements

Break , Continue

```c
#include <stdio.h>

int main()
{

    int a ;
    for ( a=0 ; a<=50 ; a++) {
        if (a % 2) continue;
        if (a == 19) break;
        printf("%d\n" , a);
    }
}

```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

Break döngüden çıkılmasını sağlar

İterative Statements

Break , Continue

```
#include <iostream>

using namespace std;

int main()
{
    int a;
    for (a=0; a<=50;a++){
        if (a == 19) break;
        if (a % 2 ) continue;
        printf("%d\n",a);
    }

}
```

```
0
2
4
6
8
10
12
14
16
18
```

Break döngüden çıkılmasını sağlar

İterative Statements

```
aaaydin-3:~ ahmetaydin$ irb
irb(main):001:0> 10.times { puts "iyi aksamlar!" }
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
iyi aksamlar!
=> 10
```

```
irb(main):002:0> list = [2, 4, 6, 8]
=> [2, 4, 6, 8]
irb(main):003:0> list.each {|value| puts value}
2
4
6
8
=> [2, 4, 6, 8]
```

Unconditional Branching

Şartsız Dallanma

- Programın calışma noktasını baska bir yere şart belirtmeden taşır.
- Program yeni yerden itibaren çalışmasına devam eder.
- **Goto komutu**
- Programın okunabilirliğini zorlastırmaktadır.

Unconditional Branching

Şartsız Dallanma

- C ve C++ da ve C# da goto komutu kullanılmaktadır.
- Java, Python, ve Ruby de **goto komutu yok !**

Unconditional Branching

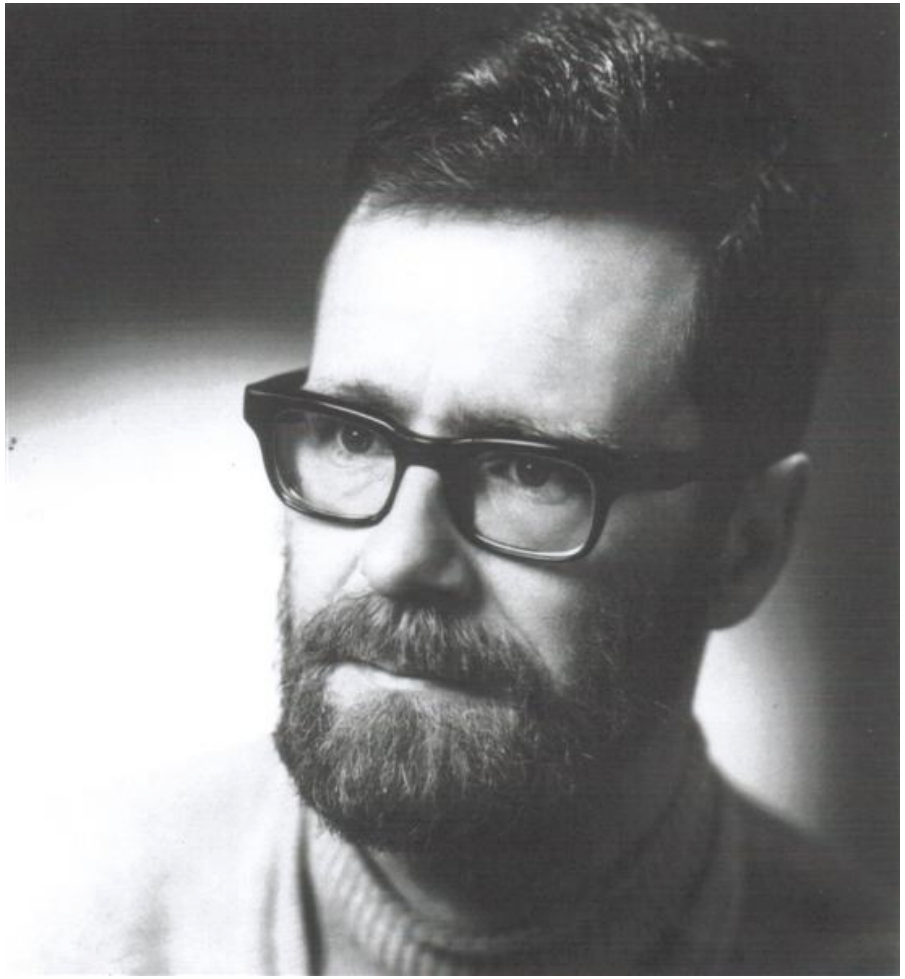
Şartsız Dallanma

- Break ve Continue gizli goto komutu kullanmaktadir.
- Döngüler gizli goto yapısı içermektedir.

Guarded Commands

Korumalı Komutlar

- **Dijkstra tarafından programların geliştirme aşamasında dogrulanması (verification) amacı için geliştirildi**
- Concurrent programlama ıcın Ada ve CSP dilleri için temel oluşturdu.



Edsger W. Dijkstra

Guarded Commands

Korumalı Komutlar

```
if <Boolean expression> -> <statement>
    [] <Boolean expression> -> <statement>
    [] ...
    [] <Boolean expression> -> <statement>
fi
```

- Bütün mantıksal ifadeleri hesaplar
- Doğru olan hiç bir değer yoksa runtime hatası verir.
- Doğru sonuçlar olduğunda rastgele seçilen bir değerle işlemi sonlandırır.

Guarded Commands

Korumalı Komutlar

```
do <Boolean expression> -> <statement>
    [] <Boolean expression> -> <statement>
    [] ...
    [] <Boolean expression> -> <statement>
od
```