Оператори за промяна на последователността

Пламен Танов
Ненко Табаков
Технологично училище „Електронни системи“
Технически университет – София

версия 0.1

Въведение

Операторите за управление на последователността на изпълнение в езика определят реда, в който се извършват изчисленията.

Един израз става оператор, когато след него се постави точка и запетая (`;`), т.е. `;` обозначава края на оператор, а не е разделител.

```
x = 0;
i++;
printf (...);
```

Съставен оператор

Фигурните скоби { и } се използват за групиране на декларации и оператори в един сложен оператор или блок, така че от синтактична гледна точка те да бъдат еквивалентни на един единствен оператор.

```
for (i = 0; i<10; i++) {
    c = getchar ();
    if (c > '0' && c < '9')
       printf ("%d", c);
}
```

Оператор if-else 1

Използва се за взимане на решения

```
if (израз)
     оператор1;
else
     оператор2;
```

или

```
if (израз)
     оператор1;
```

else частта не е задължителна

Ако **израз** е истина се изпълнява **оператор1**.
Ако не е истина се изпълнява **оператор2** (ако съществува **else** частта)

Оператор `if-else`$_2$

Понеже **`if`** операторът просто проверя числовата стойност на даден израз (дали е различна от 0, т.е. **`истина`** или е равна на 0, т.е. **`лъжа`**) са възможни няколко съкратени записи в кода

```
if (израз)
```
вместо
```
if (израз != 0)
```

```
if (!израз)
```
вместо
```
if (израз == 0)
```

Оператор if-else 3

Поради това че **else** частта не е задължителна съществува двусмислие, когато **else** бъде пропуснато от вградена последователност от оператори **if**

```
if (n > 0)
    if (a > b)
        z = a;
    else
        z = b;
```

≠

```
if (n > 0) {
    if (a > b)
        z = a;
} else
    z = b;
```

Приема се, че **else** принадлежи на най–близкия **if**, който не притежава такава част, освен в случаите когато се ползва **{ }** (примера в дясно)

Пример

```
if (n > 0)
     for (i = 0; i<n; i++)
          if (s[i] > 0) {
               printf ("%d", s[i]);
               retutrn i;
          }
else /* ГРЕШНО... този else не е към if(n > 0) */
     printf ("error – n is negative");
```

Конструкция `if-else-if`

```
if (израз1)
     оператор1;
else if (израз2)
     оператор2;
else if (израз3)
     оператор3;
else if (израз4)
     оператор4;
else
     оператор5;
```

Изразите се изчисляват поред. Ако някой от изразите е истина, то се изпълнява операторът свързан с него и с това се прекратява цялата верига.

Последната **`else`** част се изпълнява, ако никое от горните условия не е изпълнено. Тя служи като случай по подразбиране.

Пример

```
int binsearch (int x, int v[], int n) {
     int low, high, mid;

     low = 0;
     high = n -1;
     while (low <= high) {
        mid = (low + high)/2;
        if (x < v[mid])
           low = mid - 1;
        else if (x > v[mid])
           high = mid - 1;
        else
           return mid;
     }

     return -1;
}
```

Оператор switch

Използва се за изразяване на решение с много варианти (случаи). Ако даден израз съответства на някой от изброените константи се преминава през този клон на изпълнение и се продължава надолу. За излизане от **`switch`** се използва **`break;`**

`default` може да се пропусне.

```
switch (израз){
     case константен-израз :
        оператор;
     case константен-израз :
        оператор;
     case константен-израз :
        оператор;
     ...
     default : оператор
}
```

Примери

```
switch (i) {
     case 1 :
        n++;
        break;
     case 2 :
        m++;
        break;
     case 3 :
        b++;
        break;
}
```

```
switch (i) {
     case 1 :
     case 2 :
        m++;
        break;
     case 3 :
        b++;
        break;
}
```

Пример 1

```
#include <stdio.h>

int main () {
     int c,              //поредния прочетен символ
         i,              //брояч в цикъла за извеждане
         nwhite,         //брой „празни“ символи
         ndigit [10],    //колко пъти се среща дадена цифра
         nother;         //брой други символи

     nwhite = nother = 0;
     for (i = 0; i<10; i++) //инициализация на масива с броя
        ndigit[i] = 0;
```

Пример 2

```
while ((c=getchar ()) != EOF) {
        switch (c) {
            case '0' : case '1' : case '2' : //цифра
            case '3' : case '4' : case '5' :
            case '6' : case '7' : case '8' : case '9' :
                ndigit[c - '0']++;
                break;
            case ' ' :          //„празен символ“
            case '\t' :
            case '\n' :
                nwhite++;
                break;
            default :           //друг символ
                nother++;
                break;
        } //end of switch
} //end of while
```

Пример 3

```
    //извеждане на резултатите:

    printf ("digits = ");
    for (i = 0; i<10; i++)
       printf (" %d", ndigit[i]);

    printf(", white spaces=%d, other=%d\n", nwhite, nother);

}
```

Цикъл while

```
while (условие) {
    /*
          тяло, изпълнява се докато условието
  е истина (т.е. е различно от 0!)
    */
}
```

```
while (fahr <= upper) {
    celsius = 5 * (fahr - 32) / 9;
    printf ("%d\t%d\n", fahr, celsius);
    fahr = fahr + step;
}
```

Цикъл for 1

```
for (инициализация;условие;стъпка) {
    /* тяло, изпълнява се докато условието
     е истина (т.е. е различно от 0!) */
}
```

```
for (израз1; израз2; израз3) {
    оператор;
}
```

≡

```
израз1;
while (израз2) {
    оператор;
    израз3;
}
```

При ползване на **`continue`**; не са еквивалентни.
По-късно ще стане ясно защо

Цикъл for$_2$

Всяка от трите части на цикъла **for** може да бъде пропусната, но знакът точка и запетая (**;**) трябва да остане

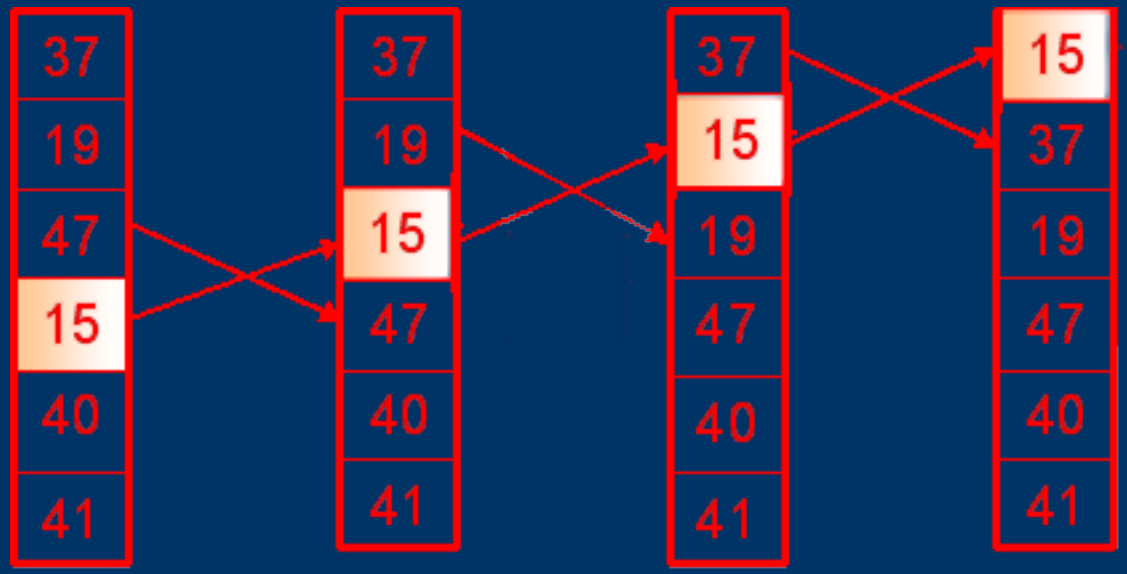
```
for (i = 0; i<=10; i += 2)
    printf (”%d\n”, i);

i = 0;
for (; i<=10; i += 2)
    printf (”%d\n”, i);

for (i = 0; i<=10; ) {
    printf (”%d\n”, i);
    i += 2;
}

for (;;) {
    //do something
}
```

Пример

```
#include <ctype.h>

int atoi (char s[]) {
     int i, n, sign;

     //търси се първият „не-празен“ символ
     for (i = 0; isspace (s[i]); i++)
     ;

     sign = (s[i] == '-') ? -1 : 1;
     if (s[i] == '+' || s[i] == '-')
        i++;
     for (n = 0; isdigit (s[i]); i++)
        n = 10 * n + (s[i] - '0');

     return sign * n;
}
```

Пример

```
void bubble_sort (int a[], int n) {
     int i, j, temp;

     for (i = 1; i < n; i++)
        for (j = n; j > i; j--)
           if (a[j] < a[j-1]) {
              //разменяме местата на j и j-1 елемент
              temp = a[j];
              a[j] = a[j-1];
              a[j-1] = temp;
           }
}
```



Цикъл do-while

```
do {
    /*
          тяло, изпълнява се докато условието
  е истина (т.е. е различно от 0!)
    */
} while (условие);
```

Цикълът **do-while** е цикъл с постусловие. При него условието се проверява накрая, след като се премине през тялото на цикъла.

Тялото винаги се изпълнява поне веднъж.

Пример

```
#include <ctype.h>

int itoa (int n, char s[]) {
     int i, sign;

     if ((sign=n)<0)
        n  = -n;
     i = 0;
     do {
        s[i++] = n % 10 + '0';
     } while (n /= 10) > 0);
     if (sign < 0)
        sign[i++] = '-';
     s[i] == '\0';
     reverse (s);
}
```

Оператори `break` и `continue`

Операторът **`break`** предоставя преждевременно излизане от циклите **`for`**, **`while`**, **`do-while`** и **`switch`**. **`break`** предизвиква внезапно прекратяване на най–вътрешния цикъл или **`switch`**.

Операторът **`continue`** предизвиква внезапно следващата итерация на най–вътрешния цикъл.

Пример

```
int trim (char s[]) {
     int n;

     for (n = strlen (s)-1; n>=0; n--)
        if (s[n] != ' ' && s[n] != '\t' && s[n] != '\n')
           break;
     s[n+1] = '\0';
     return n;
}


for (i = 0; i<n; i++) {
     if (a[i] < 0)
        continue;

     ...
}
```

Пример

```
int i = 0;
for (; i < 9; i++) {
     if (i % 2 == 0)
        continue;

...


}
```

≠

```
//ГРЕШНО !!!
int i = 0;
while (i < 9) {
     if (i % 2 == 0)
        continue;

     ...

     i++;
}
```

Оператор goto и етикети

Операторът **goto** може да се използва за безусловно предаване на управлението към оператор с етикет.

```
for (...)
    for (...)
      for (...) {
        ...
        if (disaster)
          goto error;
      }
...
error :
    //код, обработващ грешката
```