ОПЕРАТОРИ В ЕЗИКА C

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

Версия 0.4

ОПЕРАТОРИ В ЕЗИКА C

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

Един израз става оператор, когато след него се поставят точка и запетая - ;

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

ОПЕРАТОРИ В ЕЗИКА C

В **C** знакът точка и запетая обозначават край на оператор, а не е разделител.

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);
}
```

ОПЕРАТОРЪТ `if-else`

Използва се за изразяване на решения

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

`else` частта не е задължителна

Ако ***`израз`*** е истина се изпълнява ***`оператор1`***.
Ако не е истина се изпълнява ***`оператор2`***, ако съществува **`else`** частта

ОПЕРАТОРЪТ `if-else`

Понеже **`if`** операторът просто проверя числова стойност на даден израз (дали е истина или не) са възможни няколко съкратени записи в кода

```
if (израз)
```

вместо

```
if (израз != 0)
```

ОПЕРАТОРЪТ `if-else`

Понеже **`else`** частта не е задължителна съществува дусмислие, когато **`else`** бъде пропуснато от вградена последователност от опеатори **`if`**

```
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

ОПЕРАТОРЪТ `if-else`

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

```
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“);
```

КОНСТРУКЦИЯТА `else-if`

```
if (израз)
   оператор;
else if (израз)
   оператор;
else if (израз)
   оператор;
else if (израз)
   оператор;
else
   оператор;
```

Изразите се изчисляват поред. Ако някой от изразите е истина, то се изпълнява операторът свързна 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.

КОНСТРУКЦИЯТА `else-if`

```
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;
}
```

ОПЕРАТОРЪТ `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.

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

ОПЕРАТОРЪТ `switch`

```
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;
}
```

ОПЕРАТОРЪТ `switch`

```
#include <stdio.h>

int main ()
{
   int c, i, nwhite, nother, ndigit [10];

   nwhite = nother = 0;
   for (i = 0; i<10; i++)
      ndigit[i] = 0;

   while ((c=getchar ()) != EOF) {
      switch (c) {
      case '0' : case '1' : case '2' : case '3' : case '4' :
      case '5' : case '6' : case '7' : case '8' : case '9' :
         ndigit[c - '0']++;
         break;
```

ОПЕРАТОРЪТ `switch`

```
        case ' ' :
        case '\t' :
        case '\n' :
            nwhite++;
            break;
        default :
            nother++;
            break;
        }
    }
    printf („digits = “);
    for (i = 0; i<10; i++)
        printf („ %d“, ndigit[i]);
    printf („, white spaces = %d, other = %d\n“, nwhite, nother);
    }
}
```

ЦИКЪЛЪТ `while`

while (*условие*)
{
  ...
}

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

ЦИКЪЛЪТ `for`

for (*инициализация*;*условие; стъпка*)
{
  ...
}

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

еквивалентно на

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

ЦИКЪЛЪТ `for`

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 = i + 2)
   printf ("%d\n", i);

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

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

for (;;) {
}
```

ПРИМЕР

```
#include <ctype.h>

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

   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

```
#include <ctype.h>

void shellshort (int v[], int n)
{
   int gap, i, j, temp;

   for (gap = n/2; gap>0; gap/=2)
      for (i = gap; i<n; i++)
         for (j = i - gap; j>=0 && v[j]>v[j+gap]; j-=gap) {
            temp = v[j];
            v[j] = v[j+gap];
            v[j+gap] = temp;
         }
}
```

ЦИКЪЛЪТ `do-while`

```
do
        оператор
while (израз)
```

Цикълът **`do-while`** е цикъл с постусловие. При него условието се проверява накрая, след като 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);
}
```

ОПЕРАТОРИТЕ **break И continue**

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

Операторът **continue** предизвиква 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;
```

ОПЕРАТОРЪТ `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
```