Programlamaya Giriş

Akış Kontrolleri ve Döngüler

Hüseyin Ahmetoğlu

İlişkisel ve Mantıksal Operatörler

- Çoğu zaman, yapılacak eyleme karar vermeden önce iki değeri karşılaştırmanız gerekir, örneğin sayısal karşılaştırmalar.
- Her karşılaştırma işlemi iki işlenen içerir, örneğin x <= 100. Programlamada 1 <x <100 yazmak geçersizdir. Bunun yerine, iki karşılaştırma işlemini x> 1, x <100 parçalara ayırmanız ve mantıksal bir AND operatörü ile birleştirmeniz gerekir,
- (x> 1) && (x <100).

İlişkisel ve Mantıksal Operatörler

Operator	Usage	Example (x=5, y=8)
==	*expr1* == *expr2*	(x == y) → false
!=	*expr1* != *expr2*	(x != y) → true
>	*expr1* > *expr2*	(x > y) → false
>=	*expr1* >= *expr2*	(x >= 5) → true
<	*expr1* < *expr2*	(y < 8) → false
<=	*expr1* >= *expr2*	(y <= 8) → true

Operator	Description	
&&	Logical AND	*expr1* && *expr2*
||	Logical OR	*expr1* || *expr2*
!	Logical NOT	!*expr*
^	Logical XOR	*expr1* ^ *expr2*

İlişkisel ve Mantıksal Operatörler

```
// Return true if x is between 0 and 100 (inclusive)
(x >= 0) && (x <= 100)
// wrong to use 0 <= x <= 100

// Return true if x is outside 0 and 100 (inclusive)
(x < 0) || (x > 100)   //or
!((x >= 0) && (x <= 100))

// Return true if year is a leap year
// A year is a leap year if it is divisible by 4 but not by 100,
((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)
```

AND (&&)	true	false
true	true	false
false	false	false

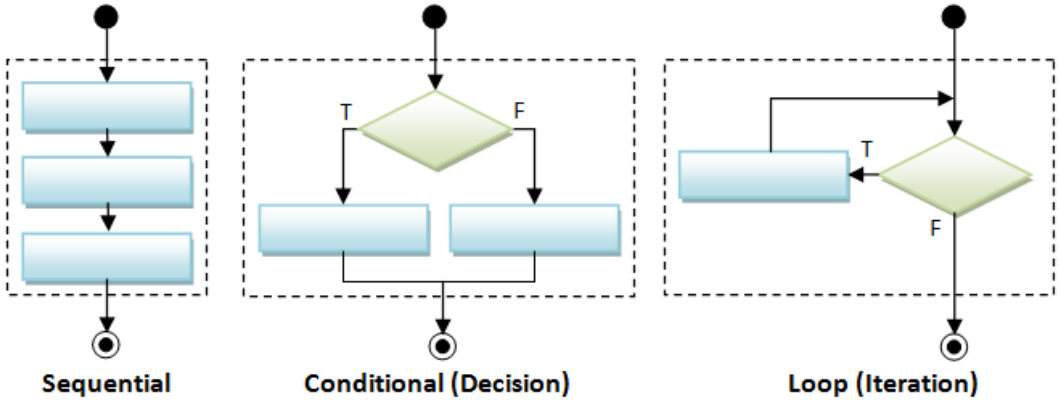
OR (||)	true	false
true	true	true
false	true	false

NOT (!)	true	false
	false	true

XOR (^)	true	false
true	false	true
false	true	false

Akış kontrolü

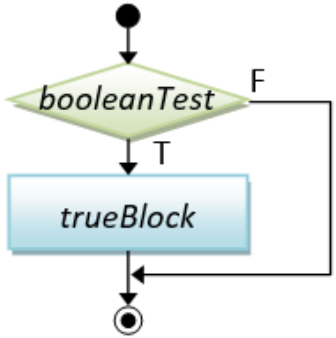
- Aşağıda gösterildiği gibi üç temel akış kontrol yapısı vardır - sıralı, koşullu (veya karar) ve döngü (veya yineleme).



if-then

```
// if-then
if ( booleanExpression ) {
   true-block ;
}
```

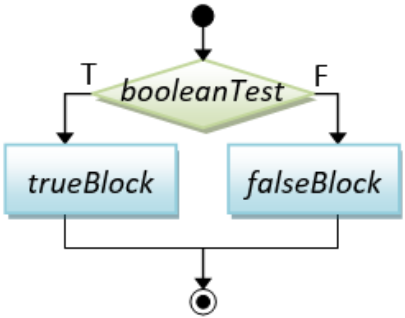
```
if (mark >= 50) {
   printf("Congratulation!\n");
   printf("Keep it up!\n");
}
```



if-then-else

```
// if-then-else
if ( booleanExpression ) {
   true-block ;
} else {
   false-block ;
}
```

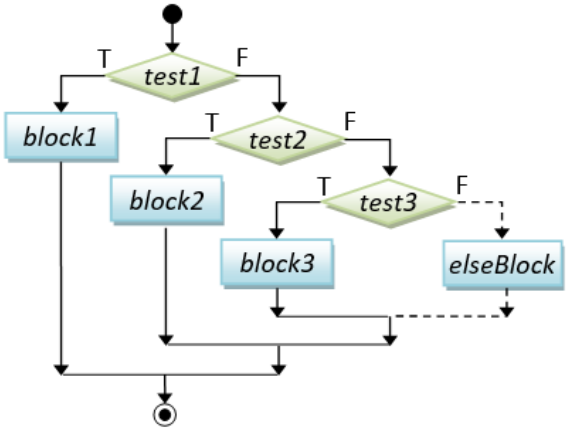
```
if (mark >= 50) {
   printf("Congratulation!\n");
   printf("Keep it up!\n");
} else {
   printf("Try Harder!\n");
}
```



nested-if

```
// nested-if
if ( booleanExpr-1 ) {
   block-1 ;
} else if ( booleanExpr-2 ) {
   block-2 ;
} else if ( booleanExpr-3 ) {
   block-3 ;
} else if ( booleanExpr-4 ) {
   ......
} else {
   elseBlock ;
}
```

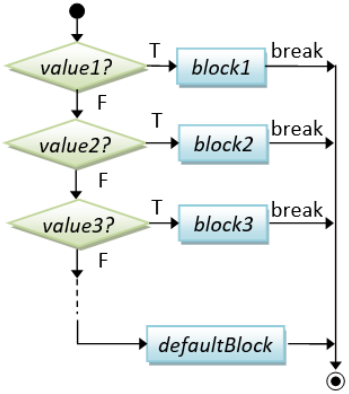
```
if (mark >= 80) {
   printf("A\n");
} else if (mark >= 70) {
   printf("B\n");
} else if (mark >= 60) {
   printf("C\n");
} else if (mark >= 50) {
   printf("D\n");
} else {
   printf("F\n");
}
```



switch-case

```
// switch-case
switch ( selector ) {
   case value-1:
      block-1; break;
   case value-2:
      block-2; break;
   case value-3:
      block-3; break;
   ......
   case value-n:
      block-n; break;
   default:
      default-block;
}
```

```
char oper; int num1, num2, result;
......
switch (oper) {
   case '+':
      result = num1 + num2; break;
   case '-':
      result = num1 - num2; break;
   case '*':
      result = num1 * num2; break;
   case '/':
      result = num1 / num2; break;
   default:
      printf("Unknown operator\n");
}
```



Koşul Operatörü

Syntax

```
booleanExpr ? trueExpr : falseExpr
```

```
printf("%s\n", (mark >= 50) ? "PASS" : "FAIL");
   // print either "PASS" or "FAIL"
max = (a > b) ? a : b;   // RHS returns a or b
abs = (a > 0) ? a : -a;  // RHS returns a or -a
```

Küme ayraçları: Blok içinde yalnızca bir ifade varsa, küme ayraçlarını {} atlayabilirsiniz.

```
if (mark >= 50)
   printf("PASS\n");   // Only one statement, can omit { } but not recommended
else {                 // more than one statements, need { }
   printf("FAIL\n");
   printf("Try Harder!\n");
}
```

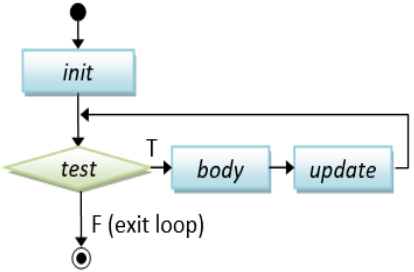
Sorular

- Kullanıcıdan alınan 1-12 arasındaki değere göre ay adını bulan programı C ile kodlayınız.
- Kullanıcıdan alınan vize ve final notlarına göre öğrencinin geçme kalma durumunu bulan programı C ile kodlayınız.
- Kullanıcıdan aldığı a, b ve c değerlerinde göre ikinci dereceden bir denklemin köklerini bulan programı C ile kodlayınız.
- Kullanıcıdan aldığı daire(d), üçgen(u), kare(k) harflerine göre geometrik şekiller için gerekli değerleri yine kullanıcıdan alıp alan ve çevre hesabı yapıp sonuçları veren programı C ile kodlayınız.

for

```
// for-loop
for (init; test; post-proc) {
   body ;
}
```

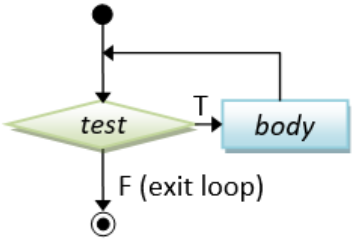
```
// Sum from 1 to 1000
int sum = 0, number;
for (number = 1; number <= 1000; ++number) {
   sum += number;
}
```



while

```
// while-do
while ( condition ) {
   body ;
}
```

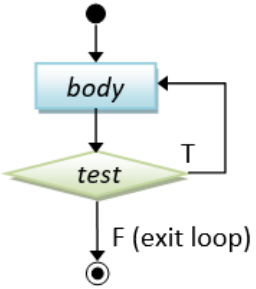
```
int sum = 0, number = 1;
while (number <= 1000) {
   sum += number;
   ++number;
}
```



do-while

```
// do-while
do {
   body ;
}
while ( condition ) ;
```

```
int sum = 0, number = 1;
do {
   sum += number;
   ++number;
} while (number <= 1000);
```



Örnek

```
// Input with validity check
bool valid = false;
int number;
do {
  // prompt user to enter an int between 1 and 10
  ......
  // if the number entered is valid, set done to exit the loop
  if (number >=1 && number <= 10) {
     valid = true;
  }
} while (!valid);   // Need a semi-colon to terminate do-while
```

```
// Game loop
bool gameOver = false;
while (!gameOver) {
   // play the game
   ......
   // Update the game state
   // Set gameOver to true if appropriate to exit the game loop
   ......
}
```

```c
/*
 * Sum from 1 to a given upperbound and compute their average (SumNumbers.c)
 */
#include <stdio.h>

int main() {
   int sum = 0;     // Store the accumulated sum
   int upperbound;

   printf("Enter the upperbound: ");
   scanf("%d", &upperbound);

   // Sum from 1 to the upperbound
   int number;
   for (number = 1; number <= upperbound; ++number) {
      sum += number;
   }
   printf("Sum is %d\n", sum);
   printf("Average is %.2lf\n", (double)sum / upperbound);

   // Sum only the odd numbers
   int count = 0;     // counts of odd numbers
   sum = 0;           // reset sum
   for (number = 1; number <= upperbound; number = number + 2) {
      ++count;
      sum += number;
   }
   printf("Sum of odd numbers is %d\n", sum);
   printf("Average is %.2lf\n", (double)sum / count);
}
```

```c
/* Prompt user for positive integers and display the count, maximum,
   minimum and average. Terminate the input with -1 (StatNumbers.c) */
#include <stdio.h>
#include <limits.h>  // for INT_MAX

int main() {
   int numberIn = 0;  // input number (positive integer)
   int count = 0;     // count of inputs, init to 0
   int sum = 0;       // sum of inputs, init to 0
   int max = 0;       // max of inputs, init to minimum
   int min = INT_MAX; // min of inputs, init to maximum (need <climits>)
   int sentinel = -1; // Input terminating value

   // Read Inputs until sentinel encountered
   printf("Enter a positive integer or %d to exit: ", sentinel);
   scanf("%d", &numberIn);
   while (numberIn != sentinel) {
      // Check input for positive integer
      if (numberIn > 0) {
         ++count;
         sum += numberIn;
         if (max < numberIn) max = numberIn;
         if (min > numberIn) min = numberIn;
      } else {
         printf("error: input must be positive! try again...\n");
      }
      printf("Enter a positive integer or %d to exit: ", sentinel);
      scanf("%d", &numberIn);
   }

   // Print result
   printf("\n");
   printf("Count is %d\n", count);
   if (count > 0) {
      printf("Maximum is %d\n", max);
      printf("Minimum is %d\n", min);
      printf("Average is %.2lf\n", (double)sum / count);
   }
}
```

break

```c
/*
 *  List primes from 1 to an upperbound (PrimeList.c).
 */
#include <stdio.h>
#include <math.h>

int main() {
   int upperbound, number, maxFactor, isPrime, factor;
   printf("Enter the upperbound: ");
   scanf("%d", &upperbound);

   for (number = 2; number <= upperbound; ++number) {
      // Not prime, if there is a factor between 2 and sqrt of number
      maxFactor = (int)sqrt(number);
      isPrime = 1;
      factor = 2;
      while (isPrime && factor <= maxFactor) {
         if (number % factor == 0) {   // Factor of number?
             isPrime = 0;
         }
         ++factor;
      }
      if (isPrime) printf("%d ", number);
   }
   printf("\n");
   return 0;
}
```

break

```c
/*
 *  List non-prime from 1 to an upperbound (NonPrimeList.c).
 */
#include <stdio.h>
#include <math.h>

int main() {
   int upperbound, number, maxFactor, factor;
   printf("Enter the upperbound: ");
   scanf("%d", &upperbound);
   for (number = 2; number <= upperbound; ++number) {
      // Not a prime, if there is a factor between 2 and sqrt(number)
      maxFactor = (int)sqrt(number);
      for (factor = 2; factor <= maxFactor; ++factor) {
         if (number % factor == 0) {   // Factor?
            printf("%d ", number);
            break;   // A factor found, no need to search for more factors
         }
      }
   }
   printf("\n");
   return 0;
}
```

continue

```
// Sum 1 to upperbound, exclude 11, 22, 33,...
int upperbound = 100;
int sum = 0;
int number;
for (number = 1; number <= upperbound; ++number) {
   if (number % 11 == 0) continue;  // Skip the rest of the loop body, continue to the next iteration
   sum += number;
}
// It is better to re-write the loop as:
for (number = 1; number <= upperbound; ++number) {
   if (number % 11 != 0) sum += number;
}
```

break and continue

```c
/* A mystery series (Mystery.c) */
#include <stdio.h>

int main() {
   int number = 1;
   while (1) {
      ++number;
      if ((number % 3) == 0) continue;
      if (number == 133) break;
      if ((number % 2) == 0) {
         number += 3;
      } else {
         number -= 3;
      }
      printf("%d ", number);
   }
   printf("\n");
   return 0;
}
```

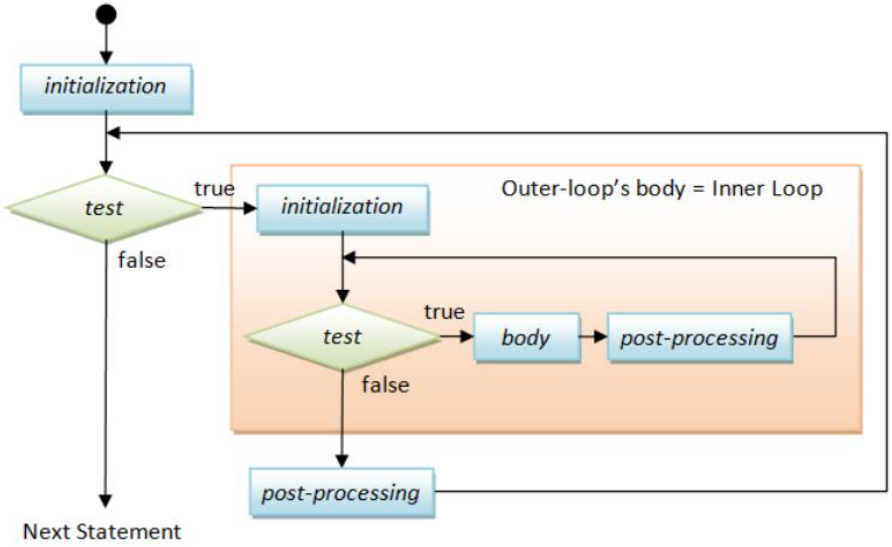
Programı sonlandırmak

- exit (): Programı sonlandırmak ve denetimi İşletim Sistemine geri döndürmek için <stdlib.h> içerisindeki exit (int exitCode) işlevini çağırabilirsiniz. Geleneksel olarak, sıfır dönüş kodu normal sonlandırmayı gösterir; sıfır olmayan bir exitCode (-1) ise anormal sonlandırmayı gösterir.
- abort (): <stdlib.h> başlığı ayrıca programı anormal bir şekilde sonlandırmak için kullanılabilen abort () adında bir işlev sağlar.
- "Return" İfadesi: Programı sonlandırmak ve denetimi İşletim Sistemine geri döndürmek için main () işlevinde bir "return returnValue" deyimi de kullanabilirsiniz.

```
if (errorCount > 10) {
   printf("too many errors\n");
   exit(-1);  // Terminate the program
              // OR abort();
}
```

```
int main() {
   ...
   if (errorCount > 10) {
      printf("too many errors\n");
      return -1;  // Terminate and return control to OS from main()
   }
   ...
}
```

İç içe döngüler

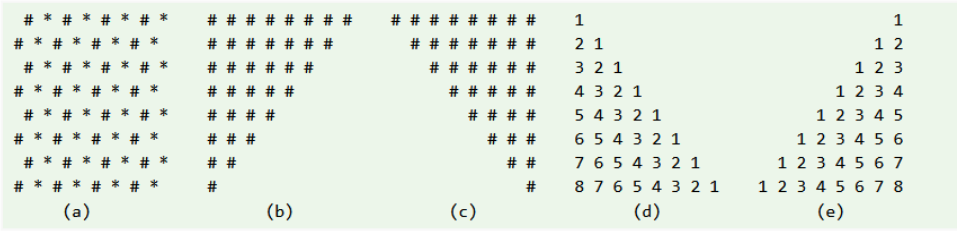


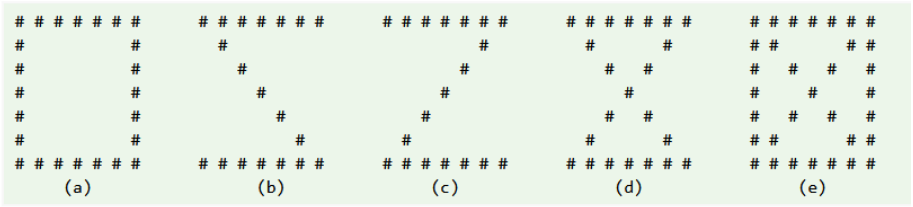
İç içe döngüler

```c
/*
 *  Print square pattern (PrintSquarePattern.c).
 */
#include <stdio.h>

int main() {
   int size = 8, row, col;
   for (row = 1; row <= size; ++row) {     // Outer loop to print all the rows
      for (col = 1; col <= size; ++col) {  // Inner loop to print all the columns of each row
         printf("# ");
      }
      printf("\n");   // A row ended, bring the cursor to the next line
   }

   return 0;
}
```

```
# # # # # # # #
# # # # # # # #
# # # # # # # #
# # # # # # # #
# # # # # # # #
# # # # # # # #
# # # # # # # #
# # # # # # # #
```

Sorular





KAYNAKLAR

- Goel, A., & Mittal, A. (2016). Computer Fundamentals and Programming in C (RMK). Pearson Education India. Retrieved from https://www.oreilly.com/library/view/computer-fundamentals-and/9789332579200
- yet another insignificant Programming Notes. (2021, April 08). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming/index.html#Cpp