

Programlamaya Giriş

Input-Output, Operatörler ve Akış Kontrolleri

Hüseyin Ahmetođlu

sizeof()

- C, işlenenin boyutunu (bayt cinsinden) elde etmek için tek bir operatör boyutu sağlar. Aşağıdaki program, temel türlerin boyutunu yazdırmak için sizeof operatörünü kullanır.

```
1  /*
2  * Print Size of Fundamental Types (SizeofTypes.cpp).
3  */
4  #include <stdio.h>
5
6  int main() {
7      printf("sizeof(char) is %d bytes.\n", sizeof(char));
8      printf("sizeof(short) is %d bytes.\n", sizeof(short));
9      printf("sizeof(int) is %d bytes.\n", sizeof(int));
10     printf("sizeof(long) is %d bytes.\n", sizeof(long));
11     printf("sizeof(long long) is %d bytes.\n", sizeof(long long));
12     printf("sizeof(float) is %d bytes.\n", sizeof(float));
13     printf("sizeof(double) is %d bytes.\n", sizeof(double));
14     printf("sizeof(long double) is %d bytes.\n", sizeof(long double));
15     return 0;
16 }
```

```
sizeof(char) is 1 bytes.
sizeof(short) is 2 bytes.
sizeof(int) is 4 bytes.
sizeof(long) is 4 bytes.
sizeof(long long) is 8 bytes.
sizeof(float) is 4 bytes.
sizeof(double) is 8 bytes.
sizeof(long double) is 12 bytes.
```

<limits.h>

```
int max = 2147483647
int min = -2147483648
unsigned int max = 4294967295
long max = 2147483647
long min = -2147483648
unsigned long max = 4294967295
long long max = 9223372036854775807
long long min = -9223372036854775808
unsigned long long max = 18446744073709551615
Bits in char = 8
char max = 127
char min = -128
signed char max = 127
signed char min = -128
unsigned char max = 255
```

```
1  /* Test integer limits in <limits.h> header */
2  #include <stdio.h>
3  #include <limits.h> // integer limits
4
5  int main() {
6      printf("int max = %d\n", INT_MAX);
7      printf("int min = %d\n", INT_MIN);
8      printf("unsigned int max = %u\n", UINT_MAX);
9
10     printf("long max = %ld\n", LONG_MAX);
11     printf("long min = %ld\n", LONG_MIN);
12     printf("unsigned long max = %lu\n", ULONG_MAX);
13
14     printf("long long max = %lld\n", LLONG_MAX);
15     printf("long long min = %lld\n", LLONG_MIN);
16     printf("unsigned long long max = %llu\n", ULLONG_MAX);
17
18     printf("Bits in char = %d\n", CHAR_BIT);
19     printf("char max = %d\n", CHAR_MAX);
20     printf("char min = %d\n", CHAR_MIN);
21     printf("signed char max = %d\n", SCHAR_MAX);
22     printf("signed char min = %d\n", SCHAR_MIN);
23     printf("unsigned char max = %u\n", UCHAR_MAX);
24     return 0;
25 }
```

<float.h>

- ▶ Benzer şekilde float.h üstbilgisi, minimum anlamlı basamak sayısı (FLT_DIG, DBL_DIG, LDBL_DIG için float, double ve long double), mantis için bit sayısı (FLT_MANT_DIG, DBL_MANT_DIG, LDBL_MANT_DIG) gibi kayan nokta sayılarının limitleri hakkında bilgi içerir.
- ▶ Siz deneyin...

printf()

- ▶ C programları, çıktıyı konsola yazdırmak için kitaplık standardının printf () işlevini kullanır. Printf () kullanmak için bir ön işlemci yönergesi "#include <stdio.h>" kullanmanız gerekir.

```
printf("Hello, world\n");
```

```
Hello, world
```

```
-
```

```
printf("Hello");  
printf(", ");  
printf("world!");  
printf("\n");  
printf("Hello\nworld\nagain\n");
```

```
Hello, world!
```

```
Hello
```

```
world
```

```
again
```

```
-
```

printf() ile çıktıyı biçilendirmek

- ▶ `FormattingString`, normal metinlerden ve dönüştürme belirtecilerinden oluşan bir dizedir. Normal metinler olduğu gibi yazdırılacaktır.
- ▶ Bir dönüşüm belirteci bir yüzde işaretiyle (%) başlar ve ardından değişkenin türünü ve çıktının biçimini (alan genişliği ve ondalık basamak sayısı gibi) belirtmek için bir kod izler. Örneğin, `%d` bir `int` anlamına gelir; Alan genişliği 3 olan bir `int` için `%3d`.
- ▶ Dönüşüm belirteçleri yer tutucular olarak kullanılır ve bunlar, biçimlendirme dizesinden sonra verilen değişkenlerle sıralı bir şekilde yazdırılır.

```
printf(formattingString, variable1, variable2, ...)
```

```

1  /*
2   * Test formatted printing for int (TestPrintfInt.c)
3   */
4  #include <stdio.h>
5
6  int main() {
7      int number1 = 12345, number2 = 678;
8      printf("Hello, number1 is %d.\n", number1);           // 1 format specifier
9      printf("number1=%d, number2=%d.\n", number1, number2); // 2 format specifiers
10     printf("number1=%8d, number2=%5d.\n", number1, number2); // Set field-widths
11     printf("number1=%08d, number2=%05d.\n", number1, number2); // Pad with zero
12     printf("number1=%-8d, number2=%-5d.\n", number1, number2); // Left-align
13     return 0;
14 }

```

```

Hello, number1 is 12345.
number1=12345, number2=678.
number1= 12345, number2= 678.
number1=00012345, number2=00678.
number1=12345  , number2=678  .

```

Dönüştürme Kodları

Type	Type Conversion Code	Type & Format
Integers	%d (or %i)	(signed) int
	%u	unsigned int
	%o	int in octal
	%x, %X	int in hexadecimal (%X uses uppercase A-F)
	%hd, %hu	short, unsigned short
	%ld, %lu	long, unsigned long
	%lld, %llu	long long, unsigned long long
Floating-point	%f	float in fixed notation
	%e, %E	float in scientific notation
	%g, %G	float in fixed/scientific notation depending on its value
	%f, %lf (printf), %lf (scanf)	double: Use %f or %lf in printf(), but %lf in scanf().
	%Lf, %Le, %LE, %Lg, %LG	long double
Character	%c	char
String	%s	string


```
int anInt = 12345;
float aFloat = 55.6677;
double aDouble = 11.2233;
char aChar = 'a';
char aStr[] = "Hello";

printf("The int is %d.\n", anInt);
//The int is 12345.
printf("The float is %f.\n", aFloat);
//The float is 55.667702.
printf("The double is %lf.\n", aDouble);
//The double is 11.223300.
printf("The char is %c.\n", aChar);
//The char is a.
printf("The string is %s.\n", aStr);
//The string is Hello.

printf("The int (in hex) is %x.\n", anInt);
//The int (in hex) is 3039.
printf("The double (in scientific) is %le.\n", aDouble);
//The double (in scientific) is 1.122330e+01.
printf("The float (in scientific) is %E.\n", aFloat);
//The float (in scientific) is 5.566770E+01.
```

```
int number = 123456;
printf("number=%d.\n", number);
// number=123456.
printf("number=%8d.\n", number);
// number= 123456.
printf("number=%3d.\n", number); // Field-width too short. Ignored.
// number=123456.
```

```
double value = 123.14159265;
printf("value=%lf;\n", value);
//value=123.141593;
printf("value=%6.2lf;\n", value);
//value=123.14;
printf("value=%9.4lf;\n", value);
//value= 123.1416;
printf("value=%3.2lf;\n", value); // Field-width too short. Ignored.
//value=123.14;
```

scanf()

- ▶ C'de, klavyeden girişleri okumak için `<stdio.h>` 'nin `scanf ()` işlevini kullanabilirsiniz. `scanf ()`, `printf ()` gibi tür dönüştürme kodunu kullanır.
- ▶ Girdiyi `scanf ()` içindeki bir değişkene yerleştirmek için, değişken adının önüne bir «ve» işareti (&) koymanız gerekir. Ve işareti (&), daha sonra açıklanacak olan operatörün adresi olarak adlandırılır.
- ▶ `double` için, `scanf ()` tür dönüştürme kodunu `%lf` kullanmanız gerekir. `Printf()` için `%f` veya `%lf` kullanabilirsiniz.

```
1  /*
2  * TestScanf.c
3  */
4  #include <stdio.h>
5
6  int main() {
7      int anInt;
8      float aFloat;
9      double aDouble;
10
11     printf("Enter an int: "); // Prompting message
12     scanf("%d", &anInt);     // Read an int from keyboard and assign to variable anInt.
13     printf("The value entered is %d.\n", anInt);
14
15     printf("Enter a floating-point number: "); // Prompting message
16     scanf("%f", &aFloat);     // Read a double from keyboard and assign to variable aFloat.
17     printf("The value entered is %f.\n", aFloat);
18
19     printf("Enter a floating-point number: "); // Prompting message
20     scanf("%lf", &aDouble);   // Read a double from keyboard and assign to variable aDouble.
21     printf("The value entered is %lf.\n", aDouble);
22
23     return 0;
24 }
```

Scanf () için Dönüş Değeri

```
int number1 = 55, number2 = 66;
int rcode = scanf("%d", &number1);
printf("return code is %d\n", rcode);
printf("number1 is %d\n", number1);
printf("number2 is %d\n", number2);
```

```
13
return code is 1
number1 is 13
number2 is 66
```

Scanf (), kullanıcı sayı1 ve sayı2'ye okunan iki tamsayı girerse 2 döndürür. Kullanıcı bir tamsayı ve ardından tamsayı olmayan bir sayı girerse 1 döndürür ve sayı2 etkilenmez. Kullanıcı tam sayı olmayan bir sayı girerse 0 döndürür ve hem sayı1 hem de sayı2 etkilenmez.

```
int number1 = 55, number2 = 66;
int rcode = scanf("%d%d", &number1, &number2);
printf("return code is %d\n", rcode);
printf("number1 is %d\n", number1);
printf("number2 is %d\n", number2);
```

```
13
25
return code is 2
number1 is 13
number2 is 25
```

Literaller

- Literaller, doğrudan bir değişkene atanabilen 123, -456, 3.14, 'a', "Merhaba" gibi belirli bir sabit değerdir.

```
int number = -123;
int sum = 4567;
int bigSum = 8234567890; // ERROR: this value is outside the range of int
```

```
int number1 = 1234; // Decimal
int number2 = 01234; // Octal 1234, Decimal 2322
int number3 = 0x1abc; // hexadecimal 1ABC, decimal 15274
int number4 = 0b10001001; // binary (may not work in some compilers)
```

Literaller

```
long number = 12345678L;    // Suffix 'L' for long
long sum = 123;            // int 123 auto-casts to long 123L
long long bigNumber = 987654321LL; // Need suffix 'LL' for long long int
```

```
float average = 55.66;    // Error! RHS is a double. Need suffix 'f' for float.
float average = 55.66f;
```

```
char letter = 'a';        // Same as 97
char anotherLetter = 98;  // Same as the letter 'b'
printf("%c\n", letter);  // 'a' printed
printf("%c\n", anotherLetter); // 'b' printed instead of the number
anotherLetter += 2;      // 100 or 'd'
printf("%c\n", anotherLetter); // 'd' printed
printf("%d\n", anotherLetter); // 100 printed
```

Literaller

```
1  /* Testing Primitive Types (TestLiteral.c) */
2  #include <stdio.h>
3
4  int main() {
5      char gender = 'm';           // char is single-quoted
6      unsigned short numChildren = 8; // [0, 255]
7      short yearOfBirth = 1945;    // [-32767, 32768]
8      unsigned int salary = 88000; // [0, 4294967295]
9      double weight = 88.88;       // With fractional part
10     float gpa = 3.88f;           // Need suffix 'f' for float
11
12     printf("Gender is %c.\n", gender);
13     printf("Number of children is %u.\n", numChildren);
14     printf("Year of birth is %d.\n", yearOfBirth);
15     printf("Salary is %u.\n", salary);
16     printf("Weight is %.2lf.\n", weight);
17     printf("GPA is %.2f.\n", gpa);
18     return 0;
19 }
```

```
Gender is m.
Number of children is 8.
Year of birth is 1945.
Salary is 88000.
Weight is 88.88.
GPA is 3.88.
```

Kaçış Karakterleri

- ▶ Yazdırılmayan karakterler ve kontrol karakterleri, ters eğik çizgi (\) ile başlayan ve ardından bir kod gelen sözde kaçış dizileri ile temsil edilebilir.

Escape Sequence	Description	Hex (Decimal)
\n	New-line (or Line-feed)	0AH (10D)
\r	Carriage-return	0DH (13D)
\t	Tab	09H (9D)
\"	Double-quote (needed to include " in double-quoted string)	22H (34D)
\'	Single-quote	27H (39D)
\\	Back-slash (to resolve ambiguity)	5CH (92D)

Aritmetik operatörler

- C sayılar için şu aritmetik operatörleri destekler: short, int, long, long long, char (8-bitlik işaretli tamsayı olarak kabul edilir), unsigned short, unsigned int, unsigned long, unsigned long long, unsigned char, float, double ve long double.

Operator	Usage	
*	<i>expr1 * expr2</i>	2 * 3 → 6; 3.3 * 1.0 → 3.3
/	<i>expr1 / expr2</i>	1 / 2 → 0; 1.0 / 2.0 → 0.5
%	<i>expr1 % expr2</i>	5 % 2 → 1; -5 % 2 → -1
+	<i>expr1 + expr2</i>	1 + 2 → 3; 1.1 + 2.2 → 3.3
-	<i>expr1 - expr2</i>	1 - 2 → -1; 1.1 - 2.2 → -1.1

Aritmetik operatörler

- ▶ operatörlerin tümü ikili operatörlerdir, yani iki işlenen alırlar. Çarpma, bölme ve kalan, toplama ve çıkarmaya göre önceliklidir. Aynı öncelik seviyesinde (örneğin, toplama ve çıkarma), ifade soldan sağa doğru değerlendirilir. Örneğin $1 + 2 + 3 - 4$, $((1 + 2) + 3) - 4$ olarak değerlendirilir.
- ▶ `int / int`'in sonucun kesildiği bir `int` ürettiğine dikkat etmek önemlidir, örneğin $1/2 \rightarrow 0$ (0.5 yerine).
- ▶ C de kuvvet operatörü (^) yoktur.

Aritmetik İfadeler

▶ $(1 + 2 * a) / 3 + (4 * (b + c) * (5 - d - e)) / f - 6 * (7 / g + h)$

$$\frac{1 + 2a}{3} + \frac{4(b + c)(5 - d - e)}{f} - 6\left(\frac{7}{g} + h\right)$$

- ▶ Matematikte olduğu gibi, "*" çarpım ve bölme "/", "+" toplama ve "-" çıkarma işlemine göre önceliklidir.
- ▶ Parantezler () daha yüksek önceliğe sahiptir.
- ▶ '+', '-', '*' ve '/' operatörleri sola bağımlıdır. Yani $1 + 2 + 3 + 4$, $((1 + 2) + 3) + 4$ olarak kabul edilir.

Karışık Tip İşlemler

- ▶ Bir aritmetik işlemin her iki işlenen de aynı türe aitse, işlem bu türde gerçekleştirilir ve sonuç o türe aittir. Örneğin, $\text{int} / \text{int} \rightarrow \text{int}$; $\text{double} / \text{double} \rightarrow \text{double}$.
- ▶ Bununla birlikte, iki işlenen farklı türlere aitse, derleyici daha küçük türün değerini daha büyük türe yükseltir (örtük tür çevirimi olarak bilinir). İşlem daha sonra daha büyük tipte gerçekleştirilir. Örneğin, $\text{int} / \text{double} \rightarrow \text{double} / \text{double} \rightarrow \text{double}$. Dolayısıyla $1/2 \rightarrow 0$, $1.0 / 2.0 \rightarrow 0.5$, $1.0 / 2 \rightarrow 0.5$, $1 / 2.0 \rightarrow 0.5$.

Type	Example	
int	2 + 3	int 2 + int 3 \rightarrow int 5
double	2.2 + 3.3	double 2.2 + double 3.3 \rightarrow double 5.5
mix	2 + 3.3	int 2 + double 3.3 \rightarrow double 2.0 + double 3.3 \rightarrow double 5.3
int	1 / 2	int 1 / int 2 \rightarrow int 0
double	1.0 / 2.0	double 1.0 / double 2.0 \rightarrow double 0.5
mix	1 / 2.0	int 1 / double 2.0 \rightarrow double 1.0 / double 2.0 \rightarrow double 0.5

Karışık Tip İşlemler

```
1  /* Testing mix-type arithmetic operations (TestMixTypeOp.c) */
2  #include <stdio.h>
3
4  int main() {
5      int i1 = 2, i2 = 4;
6      double d1 = 2.5, d2 = 5.2;
7
8      printf("%d + %d = %d\n", i1, i2, i1+i2);           // 2 + 4 = 6
9      printf("%.11f + %.11f = %.11f\n", d1, d2, d1+d2); // 2.5 + 5.2 = 7.7
10     printf("%d + %.11f = %.11f\n", i1, d2, i1+d2);    // 2 + 5.2 = 7.2 <== mix type
11
12     printf("%d / %d = %d\n", i1, i2, i1/i2);           // 2 / 4 = 0 <== NOTE: truncate
13     printf("%.11f / %.11f = %.21f\n", d1, d2, d1/d2); // 2.5 / 5.2 = 0.48
14     printf("%d / %.11f = %.21f\n", i1, d2, i1/d2);    // 2 / 5.2 = 0.38 <== mix type
15     return 0;
16 }
```

Overflow/UnderFlow

- ▶ Aritmetik işlemlerde, ortaya çıkan değer, aralığını aşarsa (yani, **Overflow/UnderFlow**) C çalışma zamanı bir hata / uyarı mesajı vermez, ancak yanlış sonuç üretir.
- ▶ **Overflow/UnderFlow** kontrolü programcının sorumluluğundadır.

```
1  /* Test Arithmetic Overflow/Underflow (TestOverflow.c) */
2  #include <stdio.h>
3
4  int main() {
5      // Range of int is [-2147483648, 2147483647]
6      int i1 = 2147483647;    // max int
7      printf("%d\n", i1 + 1); // -2147483648 (overflow)
8      printf("%d\n", i1 + 2); // -2147483647
9      printf("%d\n", i1 * i1); // 1
10
11     int i2 = -2147483648;   // min int
12     printf("%d\n", i2 - 1); // 2147483647 (underflow)
13     printf("%d\n", i2 - 2); // 2147483646
14     printf("%d\n", i2 * i2); // 0
15     return 0;
16 }
```

Bileşik Atama Operatörleri

Operator	Usage	Description	Example
=	<i>var = expr</i>	Assign the value of the LHS to the variable at the RHS	<code>x = 5;</code>
+=	<i>var += expr</i>	same as <code>var = var + expr</code>	<code>x += 5;</code> same as <code>x = x + 5</code>
-=	<i>var -= expr</i>	same as <code>var = var - expr</code>	<code>x -= 5;</code> same as <code>x = x - 5</code>
*=	<i>var *= expr</i>	same as <code>var = var * expr</code>	<code>x *= 5;</code> same as <code>x = x * 5</code>
/=	<i>var /= expr</i>	same as <code>var = var / expr</code>	<code>x /= 5;</code> same as <code>x = x / 5</code>
%=	<i>var %= expr</i>	same as <code>var = var % expr</code>	<code>x %= 5;</code> same as <code>x = x % 5</code>

Arttırma / Azaltma Operatörleri

Operator	Example	
++	x++; ++x	Increment by 1, same as x += 1
--	x--; --x	Decrement by 1, same as x -= 1

```
1  /* Test on increment (++) and decrement (--) Operator (TestIncDec.cpp) */
2  #include <stdio.h>
3
4  int main() {
5      int mark = 76;          // declare & assign
6      printf("%d\n", mark); // 76
7
8      mark++;                // increase by 1 (post-increment)
9      printf("%d\n", mark); // 77
10
11     ++mark;                // increase by 1 (pre-increment)
12     printf("%d\n", mark); // 78
13
14     mark = mark + 1;       // also increase by 1 (or mark += 1)
15     printf("%d\n", mark); // 79
16
17     mark--;                // decrease by 1 (post-decrement)
18     printf("%d\n", mark); // 78
19
20     --mark;                // decrease by 1 (pre-decrement)
21     printf("%d\n", mark); // 77
22
23     mark = mark - 1;       // also decrease by 1 (or mark -= 1)
24     printf("%d\n", mark); // 76
25     return 0;
26 }
```


Arttırma / Azaltma Operatörleri

- ▶ Arttırma / azaltma tekli operatörü, işlenenden önce (önek operatörü) veya işlenenlerden sonra (sonak operatörü) yerleştirilebilir. Anlamları farklıdır.

Operator	Description	Example	Result
++var	Pre-Increment Increment <i>var</i> , then use the new value of <i>var</i>	<code>y = ++x;</code>	same as <code>x=x+1; y=x;</code>
var++	Post-Increment Use the old value of <i>var</i> , then increment <i>var</i>	<code>y = x++;</code>	same as <code>oldX=x; x=x+1; y=oldX;</code>
--var	Pre-Decrement	<code>y = --x;</code>	same as <code>x=x-1; y=x;</code>
var--	Post-Decrement	<code>y = x--;</code>	same as <code>oldX=x; x=x-1; y=oldX;</code>

```
x = 5;
printf("%d\n", x++); // Save x (5); Increment x (=6); Print old x (5).
x = 5;
printf("%d\n", ++x); // Increment x (=6); Print x (6).
// This is confusing! Try to avoid! What is i=++i? What is i=i++?
```

Implicit Type-Conversion vs. Explicit Type-Casting

- ▶ Bir değerin bir türden başka bir türe dönüştürülmesine, tür dönüştürme (*type casting* or *type conversion*) denir. İki tür *type casting* vardır:
- ▶ Derleyici tarafından otomatik olarak gerçekleştirilen örtük tür dönüştürme (Implicit type-conversion)
- ▶ İşlenen biçiminde tek tip çevrim operatörü (*new-type*) aracılığıyla açık tip çevrimi (Explicit type-casting).

Implicit (Automatic) Type Conversion

- ▶ Başka bir temel türdeki değişkene temel (yerleşik) türden bir değer atadığınızda, iki tür uyumluysa, C değeri otomatik olarak alıcı türe dönüştürür. Örneğin,
- ▶ Bir double değişkene bir int değeri atarsanız, derleyici int değerini otomatik olarak double'a çevirir (ör. 1'den 1.0'a) ve bunu double değişkene atar.
- ▶ Bir int değişkenine bir double değeri atarsanız, derleyici double değeri otomatik olarak bir int değerine çevirir (ör. 1,2'den 1'e) ve bunu int değişkenine atar. Kesirli kısım kesilir ve kaybolur.
- ▶ Bazı derleyiciler bir uyarı / hata "olası hassasiyet kaybı" verir.
- ▶ İki tür uyumlu değilse, C otomatik tür dönüştürme gerçekleştirmeyecektir.

Implicit (Automatic) Type Conversion

```
1  /*
2   * Test implicit type casting (TestImplicitTypeCast.c)
3   */
4  #include <stdio.h>
5
6  int main() {
7      int i;
8      double d;
9
10     i = 3;
11     d = i;    // Assign an int value to double
12     printf("d = %lf\n", d); // d = 3.0
13
14     d = 5.5;
15     i = d;    // Assign a double value to int
16     printf("i = %d\n", i); // i = 5 (truncated, no warning!)
17
18     i = 6.6; // Assign a double literal to int
19     printf("i = %d\n", i); // i = 6 (truncated, no warning!)
20 }
```

Explicit Type-Casting

- ▶ Tip çevrimini (*new-type*) işlenen biçiminde tekli tip çevrim operatörü ile açıkça gerçekleştirebilirsiniz. Tür atama operatörü, bir türden işlenen alır ve yeni türde eşdeğer bir değer döndürür.

```
printf("%lf\n", (double)5); // int 5 -> double 5.0
printf("%d\n", (int)5.5); // double 5.5 -> int 5

double aDouble = 5.6;
int anInt = (int)aDouble; // return 5 and assign to anInt. aDouble does not change!
```

Explicit Type-Casting

```
1  /*
2   * Converting between Celsius and Fahrenheit (ConvertTemperature.c)
3   *   Celsius = (5/9)(Fahrenheit-32)
4   *   Fahrenheit = (9/5)Celsius+32
5   */
6  #include <stdio.h>
7
8  int main() {
9      double celsius, fahrenheit;
10
11     printf("Enter the temperature in celsius: ");
12     scanf("%lf", &celsius);
13     fahrenheit = celsius * 9 / 5 + 32;
14     // 9/5*celsius + 32 gives wrong answer! Why?
15     printf("%.2lf degree C is %.2lf degree F\n", celsius, fahrenheit);
16
17     printf("Enter the temperature in fahrenheit: ");
18     scanf("%lf", &fahrenheit);
19     celsius = (fahrenheit - 32) * 5 / 9;
20     // 5/9*(fahrenheit - 32) gives wrong answer! Why?
21     printf("%.2lf degree F is %.2lf degree C\n", fahrenheit, celsius);
22     return 0;
23 }
```

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 $x \leq 100$. Programlamada $1 < x < 100$ yazmak geçersizdir. Bunun yerine, iki karşılaştırma işlemi $x > 1$, $x < 100$ 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)
==	<i>expr1 == expr2</i>	(x == y) → false
!=	<i>expr1 != expr2</i>	(x != y) → true
>	<i>expr1 > expr2</i>	(x > y) → false
>=	<i>expr1 >= expr2</i>	(x >= 5) → true
<	<i>expr1 < expr2</i>	(y < 8) → false
<=	<i>expr1 <= expr2</i>	(y <= 8) → true

Operator	Description	
&&	Logical AND	<i>expr1 && expr2</i>
	Logical OR	<i>expr1 expr2</i>
!	Logical NOT	<i>!expr</i>
^	Logical XOR	<i>expr1 ^ expr2</i>

İlişkisel ve Mantıksal Operatörler

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

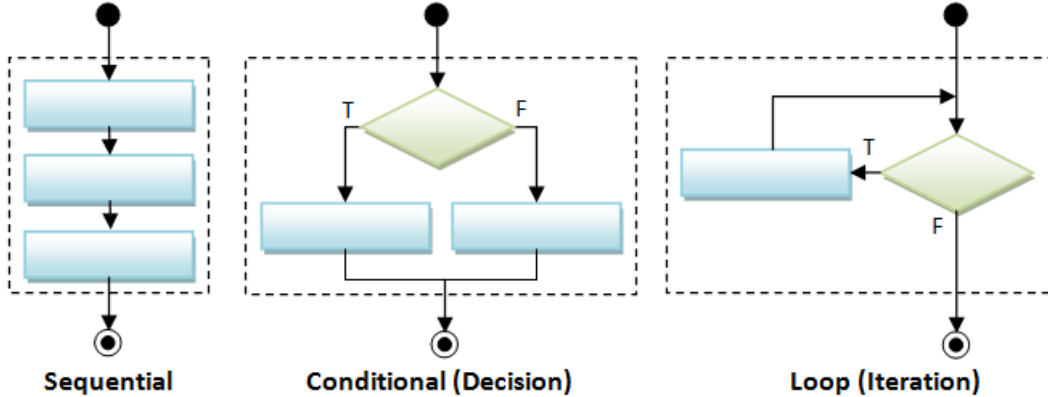
// Return true if x is outside 0 and 100 (inclusive)
(x < 0) || (x > 100) //or
!((x >= 0) && (x <= 100))

// 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

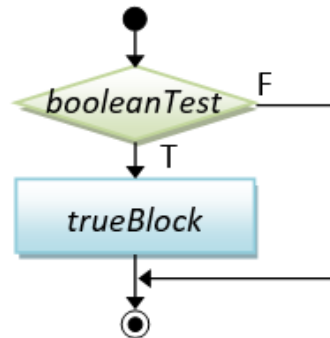
- Aşağıda gösterildiği gibi üç temel akış kontrol yapısı vardır - sıralı, koşullu (veya karar) ve döngü (veya yineleme).



if-then

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

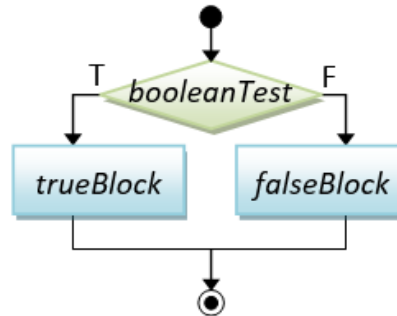
```
if (mark >= 50) {  
    printf("Congratulation!\n");  
    printf("Keep it up!\n");  
}
```



if-then-else

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

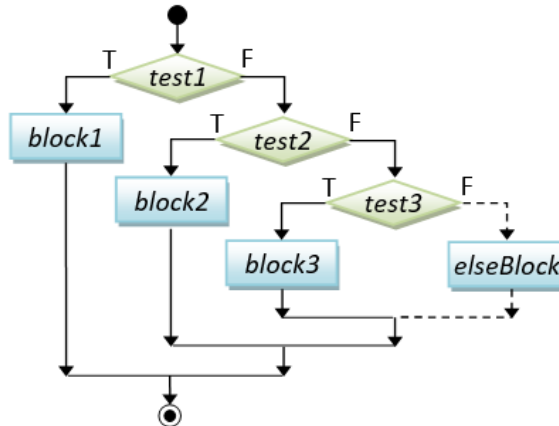
```
if (mark >= 50) {
    printf("Congratulation!\n");
    printf("Keep it up!\n");
} else {
    printf("Try Harder!\n");
}
```



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 ;
}
}
```

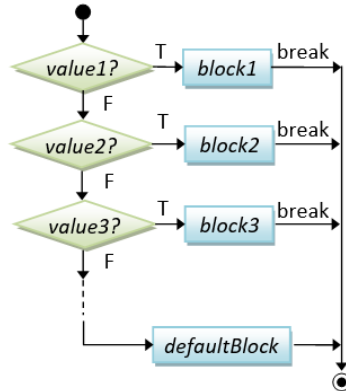
```
if (mark >= 80) {
    printf("A\n");
} else if (mark >= 70) {
    printf("B\n");
} else if (mark >= 60) {
    printf("C\n");
} else if (mark >= 50) {
    printf("D\n");
} else {
    printf("F\n");
}
}
```



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.

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>