Programlamaya Giriş

Input-Output, Aritmetiksel Operatörler

Hüseyin Ahmetoğlu

sizeof()

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

```c
/*
 * Print Size of Fundamental Types (SizeofTypes.cpp).
 */
#include <stdio.h>

int main() {
   printf("sizeof(char) is %d bytes.\n", sizeof(char));
   printf("sizeof(short) is %d bytes.\n", sizeof(short));
   printf("sizeof(int) is %d bytes.\n", sizeof(int));
   printf("sizeof(long) is %d bytes.\n", sizeof(long));
   printf("sizeof(long long) is %d bytes.\n", sizeof(long long));
   printf("sizeof(float) is %d bytes.\n", sizeof(float));
   printf("sizeof(double) is %d bytes.\n", sizeof(double));
   printf("sizeof(long double) is %d bytes.\n", sizeof(long double));
   return 0;
}
```

```
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
```

```c
/* Test integer limits in <limits.h> header */
#include <stdio.h>
#include <limits.h>   // integer limits

int main() {
   printf("int max = %d\n", INT_MAX);
   printf("int min = %d\n", INT_MIN);
   printf("unsigned int max = %u\n", UINT_MAX);

   printf("long max = %ld\n", LONG_MAX);
   printf("long min = %ld\n", LONG_MIN);
   printf("unsigned long max = %lu\n", ULONG_MAX);

   printf("long long max = %lld\n", LLONG_MAX);
   printf("long long min = %lld\n", LLONG_MIN);
   printf("unsigned long long max = %llu\n", ULLONG_MAX);

   printf("Bits in char = %d\n", CHAR_BIT);
   printf("char max = %d\n", CHAR_MAX);
   printf("char min = %d\n", CHAR_MIN);
   printf("signed char max = %d\n", SCHAR_MAX);
   printf("signed char min = %d\n", SCHAR_MIN);
   printf("unsigned char max = %u\n", UCHAR_MAX);
   return 0;
}
```

<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>" kuallanmanız 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 belirticilerinden oluşan bir dizedir. Normal metinler olduğu gibi yazdırılacaktır.
- Bir dönüşüm belirticisi 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, ...)
```

```c
/*
 * Test formatted printing for int (TestPrintfInt.c)
 */
#include <stdio.h>

int main() {
   int number1 = 12345, number2 = 678;
   printf("Hello, number1 is %d.\n", number1);                // 1 format specifier
   printf("number1=%d, number2=%d.\n", number1, number2);     // 2 format specifiers
   printf("number1=%8d, number2=%5d.\n", number1, number2);   // Set field-widths
   printf("number1=%08d, number2=%05d.\n", number1, number2); // Pad with zero
   printf("number1=%-8d, number2=%-5d.\n", number1, number2); // Left-align
   return 0;
}
```

```
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.

```c
/*
 * TestScanf.c
 */
#include <stdio.h>

int main() {
   int anInt;
   float aFloat;
   double aDouble;

   printf("Enter an int: ");  // Prompting message
   scanf("%d", &anInt);       // Read an int from keyboard and assign to variable anInt.
   printf("The value entered is %d.\n", anInt);

   printf("Enter a floating-point number: ");  // Prompting message
   scanf("%f", &aFloat);     // Read a double from keyboard and assign to variable aFloat.
   printf("The value entered is %f.\n", aFloat);

   printf("Enter a floating-point number: ");  // Prompting message
   scanf("%lf", &aDouble);     // Read a double from keyboard and assign to variable aDouble.
   printf("The value entered is %lf.\n", aDouble);

   return 0;
}
```

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

```c
/* Testing Primitive Types (TestLiteral.c) */
#include <stdio.h>

int main() {
   char gender = 'm';              // char is single-quoted
   unsigned short numChildren = 8; // [0, 255]
   short yearOfBirth = 1945;       // [-32767, 32768]
   unsigned int salary = 88000;    // [0, 4294967295]
   double weight = 88.88;          // With fractional part
   float gpa = 3.88f;              // Need suffix 'f' for float

   printf("Gender is %c.\n", gender);
   printf("Number of children is %u.\n", numChildren);
   printf("Year of birth is %d.\n", yearOfBirth);
   printf("Salary is %u.\n", salary);
   printf("Weight is %.2lf.\n", weight);
   printf("GPA is %.2f.\n", gpa);
   return 0;
}
```

```
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

- Yazdırılamayan 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	
*	*expr1* * *expr2*	2 * 3 → 6; 3.3 * 1.0 → 3.3
/	*expr1* / *expr2*	1 / 2 → 0; 1.0 / 2.0 → 0.5
%	*expr1* % *expr2*	5 % 2 → 1; -5 % 2 → -1
+	*expr1* + *expr2*	1 + 2 → 3; 1.1 + 2.2 → 3.3
-	*expr1* - *expr2*	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.
- İnt / int'in sonucun kesildiği bir int ürettiğine dikkat etmek önemlidir, örneğin 1/2 → 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, int / int → int; double / double → 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, int / double → double / double → double. Dolayısıyla 1/2 → 0, 1.0 / 2.0 → 0.5, 1.0 / 2 → 0.5, 1 / 2.0 → 0.5.

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

Karışık Tip İşlemler

```c
/* Testing mix-type arithmetic operations (TestMixTypeOp.c) */
#include <stdio.h>

int main() {
   int i1 = 2, i2 = 4;
   double d1 = 2.5, d2 = 5.2;

   printf("%d + %d = %d\n", i1, i2, i1+i2);          // 2 + 4 = 6
   printf("%.1lf + %.1lf = %.1lf\n", d1, d2, d1+d2); // 2.5 + 5.2 = 7.7
   printf("%d + %.1lf = %.1lf\n", i1, d2, i1+d2);    // 2 + 5.2 = 7.2  <== mix type

   printf("%d / %d = %d\n", i1, i2, i1/i2);          // 2 / 4 = 0  <== NOTE: truncate
   printf("%.1lf / %.1lf = %.2lf\n", d1, d2, d1/d2); // 2.5 / 5.2 = 0.48
   printf("%d / %.1lf = %.2lf\n", i1, d2, i1/d2);    // 2 / 5.2 = 0.38  <== mix type
   return 0;
}
```

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 soumluluğundadır.

```c
/* Test Arithmetic Overflow/Underflow (TestOverflow.c) */
#include <stdio.h>

int main() {
   // Range of int is [-2147483648, 2147483647]
   int i1 = 2147483647;      // max int
   printf("%d\n", i1 + 1);   // -2147483648 (overflow)
   printf("%d\n", i1 + 2);   // -2147483647
   printf("%d\n", i1 * i1);  // 1

   int i2 = -2147483648;     // min int
   printf("%d\n", i2 - 1);   // 2147483647 (underflow)
   printf("%d\n", i2 - 2);   // 2147483646
   printf("%d\n", i2 * i2);  // 0
   return 0;
}
```

Bileşik Atama Operatörleri

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

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 (sonek operatörü) yerleştirilebilir. Anlamları farklıdır.

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

```
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

```c
/*
 * Test implicit type casting (TestImplicitTypeCast.c)
 */
#include <stdio.h>

int main() {
   int i;
   double d;

   i = 3;
   d = i;    // Assign an int value to double
   printf("d = %lf\n", d);  // d = 3.0

   d = 5.5;
   i = d;    // Assign a double value to int
   printf("i = %d\n", i);  // i = 5 (truncated, no warning!)

   i = 6.6;  // Assign a double literal to int
   printf("i = %d\n", i);  // i = 6 (truncated, no warning!)
}
```

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

```c
/*
 * Converting between Celsius and Fahrenheit (ConvertTemperature.c)
 *   Celsius = (5/9)(FahrenheitB32)
 *   Fahrenheit = (9/5)Celsius+32
 */
#include <stdio.h>

int main() {
   double celsius, fahrenheit;

   printf("Enter the temperature in celsius: ");
   scanf("%lf", &celsius);
   fahrenheit = celsius * 9 / 5 + 32;
   // 9/5*celsius + 32 gives wrong answer! Why?
   printf("%.2lf degree C is %.2lf degree F\n", celsius, fahrenheit);

   printf("Enter the temperature in fahrenheit: ");
   scanf("%lf", &fahrenheit);
   celsius =  (fahrenheit - 32) * 5 / 9;
   // 5/9*(fahrenheit - 32) gives wrong answer! Why?
   printf("%.2lf degree F is %.2lf degree C\n", fahrenheit, celsius);
   return 0;
}
```

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