**Farklı Türlerin Birbirine Atanması (Dönüştürülmesi)**

Bir atama işleminde kaynak (source) ve hedef (destination) türler vardır. En normal durum bu türlerin aynı olmasıdır. Atama işleminde kaynak tür derleyici tarafından önce otomatik olarak (implicit) hedef türe dönüştürülür. Daha sonra atama yapılır. Yani T1 türünden T2 türüne doğrudan atama işlemi yapılabilmesi için T1 türünden T2 türüne otomatik dönüştürmenin (implicit conversion) var olması gerekir. C’de neredeyse her tür her türe atanabilir. Bu her ne kadar saçma görülse de mümkündür. Örneğin Java’da long türünden int türüne doğrudan dönüşüm geçersizdir. C’de bu durum yalnızca bir warninge neden olur. Bu da potansiyel bir bilgi kaybı olabilir endişesidir. Çünkü bilindiği gibi long türü int türünün tutamadığı değer tutan 8 bytelık bir tamsayı türüdür. Bu yüzden atama durumunda bilgi kaybı oluşabilir. Veri kaybı oluşacağı ise kesindir. Aşağıdaki örnekte sabit int sınırları içerisinde olduğu için sadece veri kaybı oluşur, bilgi kaybı oluşmaz:

#include <stdio.h>

int main(void)

{

long a = 10;

int b = a; //warning

}

Bu warning genellikle şöyle bir warning mesajıdır: Narrowing conversion from long to int. Yani uyarı mesajından da anlaşılabileceği gibi burda bir daraltıcı dönüşüm(narrowing conversion) vardır. Bir de bunun tersi olan genişletici dönüşüm(widing conversion) vardır. Tahmin edebileceği gibi bu da küçük türlerden büyük türlere yapılan atamalardır. Bunlar çoğu programlama dilinde syntax error’e sebep olmaz iken narrowing conversion riskli olması sebebiyle bazı dillerde syntax error’e sebep olur. Widing conversion yani genişletici dönüşümlerin pek çok dilde sentaks hatasına neden olmaması bilgi kaybına yol açmayacak olmalarından kaynaklandığını söyleyebiliriz. C’de widing conversionlar warninge bile sebep olmayıp oldukça doğal bir durumdur. Örneğin aşağıdaki örnekte olduğu gibi int türünden bir değişkenin long türünden bir değişkene atanması gibi :

int main(void)

{

int a = 45;

long b = a;

}

1. Short türünden bir değişken int türüne atanırken integral promotion kuralı devreye girer. Bu atamada bir warning söz konusu değildir.

int main(void)

{

short a = 56;

int b = a;

}

2. Küçük tamsayı türünden büyük tamsayı türüne atama yapıldığında küçük tamsayı içerisindeki değer pozitifse sayının genişletilen (yüksek anlamlı) bitleri 0(sıfır) ile beslenir. Örneğin:

#include <stdio.h>

int main(void)

{

short a = 0x11FC;

printf("a : %d\n",a);

int b = a;

printf("b :0x%032x\n",b);

printf("b : %d\n",b);

}

3. İşaretli tamsayı türlerinden işaretsiz tamsayı türleri atama durumları eğer işaretli türün değeri negatif ise bu durumda bir undefined behaviour(Tanımsız Davranış) oluşur.

#include <stdio.h>

int main(void)

{

unsigned int a;

int b;

b= -10;

a = b;

printf("%u\n",a);

}

4. char türünden int türüne doğrudan dönüşüm C’de vardır. Bu durumda char türündeki değişkene eğer UNICODE karakter ile ilk değer verilmişse bu durumda ASCII tablosunda o karaktere denk gelen değer int türden değişkene atanır. Örneğin:

#include <stdio.h>

int main(void)

{

char ch = 'A';

int a = ch;

printf("%d\n", a);

}

Bu kodun ekran çıktısı ‘A’ karakterinin ASCII tablosunda 65 sayısına denk gelmesinden dolayı 65 olarak elde edilecektir.

5. C’de int türünden char türüne doğrudan dönüşüm geçerlidir. Bu durumda char sınırlarının aşılması durumunda yüksek anlamlı byte değerleri atılır. Kısacası veri kaybı olacağı kesin iken bilgi kaybının oluşması senaryoya bağlıdır. Örneğin aşağıdaki örnekte int sınırları aşılmadığı için bilgi kaybı oluşmaz ve ekrana başarılı bir şekilde 10 sabiti yazdırılır:

#include <stdio.h>

int main(void)

{

char ch ;

int a = 10;

ch = a;

printf("%d\n",ch);

}

6. C’de char türünden short türüne doğrudan dönüşüm geçerlidir. Örneğin:

#include <stdio.h>

int main(void)

{

char ch = 'a';

short sVal = ch;

printf("%d\n",sVal);

}

7. Gerçek sayı türlerinden tamsayı türlerine doğrudan atama(Standard Conversion) geçerlidir. Ancak bu durumda örneğin double türünden int türüne yapılan bir atama narrowing conversion söz konusu olacağından bununla ilgili bir uyarı mesajı verilir:

#include <stdio.h>

int main(void)

{

int a = 10;

double dVal = 12.5;

a = dVal;

printf("%d\n",a);

}

8. Aşağıdaki örnekte bir narrowing conversion söz konusudur. Bunun nedeni sabitler konusundan hatırlamamız gereken bir kuralla ilgilidir. (Sayı nokta içeriyorsa ve sonuna bir almışsa double türdendir.) Bu örnek ile anlatılmak istenen diğer bir şey de gerçek sayılarda da narrowing conversion bir uyarı mesajına neden olmaktadır:

#include <stdio.h>

int main(void)

{

int a = 10;

float dVal = 20.9;

}

**İşlem Öncesi Otomatik Tür Dönüşümleri**

Programlama dillerinde yalnızca değişkenlerin ve sabitlerin değil aynı zamanda her ifadenin de bir türü vardır. Örneğin a ‘nın bir türü vardır, 10 sabitinin de bir türü bardır , a + 10 ifadesinin de bir türü vardır. Bir operatörün operandları farklı türlerden olabilir. Derleyici bir operatörle karşılaştığında operandların türlerine bakar. Bunlar farklı türlerdense önce onları aynı türe dönüştürür, ondan sonra işlemi yapar. Özet kural şudur:

“Küçük tür büyük türe dönüştürülür ve sonuç büyük tür türünden çıkar”

Örneğin:

int a = 10;

double b = 2.5;

int c;

c = a + b //işlem double türünden yapılır.

Burada a + b işleminden önce a double türüne dönüştürülür, sonra işlem yapılır. Sonuç double türünden çıkar. En önemli kurallardan bir tane eğer operandlardan biri bile double türden ise bu durumda işlem her zaman double türünden yapılır. Ancak atama yapılırken double türünden int türüne atama yapılacağından sayının noktadan sonraki kısmı atılır.

“Küçük tür büyük türe dönüştürülür ve sonuç büyük tür türünden çıkar” kuralının da bazı ayrıntıları vardır:

1. İşlem öncesinde her iki operand da short ya da char türündense (yani int türünden küçükse) önce her iki operand da bağımsız olarak int türüne dönüştürülür, sonuç int türünden çıkar. Bu kurala int türüne yükseltme (integral promotion) denilmektedir. Örneğin:

#include <stdio.h>

int main(void)

{

short s1,s2;

short result;

s1 = 10;

s2 = 20;

result = s1 + s2;

printf("result : %d\n",result);

}

2. Gerçek sayı türleriyle tamsayı türleri işleme sokuluyorsa dönüştürme her zaman gerçek sayı türlerine doğru yapılır. Sonuç dönüştürülen gerçek sayı türünden çıkar. Örneğin:

#include <stdio.h>

int main(void)

{

int a;

double b;

double result;

a = 10;

b = 20.456;

result = a + b;

printf("result : %lf\n",result);

}

3. Bölme işleminde her iki operand da tamsayı türlerine ilişkinse sonuç tamsayı türünden çıkar. Bölme işlemi yapılır sonucun noktadan sonraki kısmı atılır. Örneğin:

#include <stdio.h>

int main(void)

{

int a = 10;

int b = 3;

double result = a / b; //noktadan sonraki kısım atılır!

printf("result : %lf\n",result);

}

4. char türü diğer türlerle işleme sokulabilir. Örneğin:

#include <stdio.h>

int main(void)

{

char ch = 'a';

int b = 15;

int val = ch + b;

printf("val = %d\n",val);

}

5. İşaretsiz tamsayı türleriyle işaretli tamsayı türleri işleme sokulursa işlem işaretsiz tamsayı türünden yapılır. Aynı aşağıdaki örnekte olduğu gibi:

#include <stdio.h>

int main(void)

{

int a=-100;

unsigned b=20;

printf("%u",a+b);

return 0;

}

**Anahtar Notlar :** Aşağıdaki örnekte ekrana hayır yanlış basılacaktır. Bunun nedeni char türünün maksimum değerinin C’de 127 olmasından kaynaklanmaktadır. 178 sayısının yüksek anlamlı byte değerleri atılır ve son durumda farklı bir değer elde edilir. Bu sebeple ifin false bölümü çalıştırılır.

#include <stdio.h>

int main(void) {

char ch = 178;

if (ch == 178)

printf("Evet dogru");

else

printf("Hayir yanlis");

}

**Anahtar Notlar :** İşlem öncesinde küçük türün büyük türe dönüştürülmesi geçici değişken yoluyla yapılmaktadır. Önce büyük tür türünden geçici bir değişken yaratılır ve küçük türün içerisindeki değer ona atanır. İşleme geçici değişken sokulur, işlem sonrasında geçici değişken yok edilir.

**Tür Dönüştürme Operatörü(Type Casting Operator)**

Tür dönüştürme operatörü tek operandlı örnek durumunda bir operatördür. Genel biçimi şöyledir:

(tür) operand

Buradaki parantez operatörü belirtmektedir. Öncelik parantezi değildir. Tür dönüştürme operatörü öncelik tablosunun ikinci öncelik seviyesinde bulunan sağdan sola doğru yapılan unary(tek operand alan) bir operatördür. Örneğin:

a = (double) b / c;

i1 : (double) b

i2 : i1 / c

a = i2

Dönüştürme işlemi geçici değişken yoluyla yapılmaktadır. Yani önce dönüştürülecek tür türünden geçici bir değişken yaratılır, dönüştürülecek değer ona atanır, sonra işleme geçici değişken sokulur. İşlem sonrasında geçici değişken yok edilir.

Tür dönüştürme operatörü ile yapılan dönüştürmelere “explicit conversion” denilmektedir. Bilindiği gibi otomatik tür dönüştürmelere “implicit conversion” denilmektedir. Tür dönüştürme operatörü ile her temel tür her temel türe dönüştürülebilir. Bu doğrudan dönüşümde warning olan atamalar warning olmadan gerçekleştirilir. Örneğin:

#include <stdio.h>

int main(void) {

int a;

short b;

a = 1000000;

b = (short)a;

printf("b = %d\n", b);

}

Şüphesiz eğer burada tür dönüştürme operatörü burada kullanılmasaydı, bilgi kaybından dolayı warning oluşacaktı. Ancak tür dönüştürme kullanıldığı için warning kaldırılmış oldu. Aşağıdai maddeler else if gibi ele alınmalıdır:

1. Eğer dönüştürme sırasında kaynak türün belirttiği değer hedef türün sınırları içerisinde kalıyorsa bilgi kaybı oluşmaz. Örneğin:

#include <stdio.h>

int main(void)

{

long a;

int b;

a = 100;

b =(int)a;

printf("b = %d\n", b);

}

2. Büyük tamsayı türünden küçük tamsayı türüne dönüştürme yapıldığından sayının yüksek anlamlı byte değerleri atılır. Böylece sayı alakasız bir durum gelebilir. Örneğin:

#include <stdio.h>

int main()

{

int a;

short b;

a = 10000000;

b = (short)a;

printf("a = %X\n",a);

printf("b = %X\n",b);

}

Bu örnekte değer short sınırları içerisinde olmadığından dolayı bilgi kaybı oluşur.

3.short türünden char türüne dönüştürme yapıldığında sayının işareti ne olursa olsun sayının bitleri işaretsiz olarak yorumlanır. Örneğin:

#include <stdio.h>

int main()

{

short a;

char ch;

a = 97;

ch = (char)a;

printf("ch = %c\n",ch);//ch = a

}

Örneğin:

#include <stdio.h>

int main()

{

short a;

char ch;

a = -130;

ch = (char)a;

printf("ch = %c\n",ch);

}

Bu örnekte ise sayı char sınırlarının dışında olduğu için sayının yüksek anlamlı byte değerleri atılır. Son durumda kalan bytelarla UNICODE tablodaki değeri alınır.

4. double türünden float türüne tür dönüştürme operatörüyle ile dönüştürme yapıldığında bilgi kaybı oluşabilir. Bu durumda IEEE 754 formatının yuvarlama kuralları uygulanarak float türünde gösterilebilecek en yakın sayı elde edilir. Örneğin:

#include <stdio.h>

int main()

{

double a;

float b;

a = 100.456789;

b = (float)a;

printf("a = %lf\n",a);//a = 100.456789

printf("b = %f\n",b);//b = 100.456787

}

5. Gerçek sayı türünden tamsayı türüne tür dönüştürme operatörü ile yapılan dönüştürme işlemi iki aşamada gerçekleştirilir: Birinci aşamada hedef tür long türü dışında işaretli bir tamsayı türünde ise (char, short veya int) önce int türüne dönüştürülür. Hedef tür long ise long türüne dönüştürülür. Bu dönüştürme sırasında sayının noktadan sonraki kısmı atılır. Bu durumda elde edilen değer dönüştürülecek türün(long veya int) sınırları içerisinde kalıyorsa bir sorun oluşmaz. Eğer elde edilen değer dönüştürülecek türün (long veya int) sınırları içerisinde kalmıyorsa ve sayı negatifse dönüştürülecek türün (long veya int) tutabildiği en küçük değer alınır, sayı pozitifse dönüştürülecek türün (long veya int) tutabildiği en büyük değer alınır. İkinci aşamada hedef tür int veya long ise değer birinci aşamada elde edilen değerdir. Hedef tür bytei short ya da char türlerinden biriyse elde edilen değer birinci aşamada elde edilen değerin yüksek anlamlı byte değerleri atılmış biçimidir. Örneğin:

#include <stdio.h>

int main()

{

double a;

int b;

a = 100.456789;

b = (int)a;

printf("b = %d\n",b);

}

Burada sayının noktadan sonraki kısmı atıldığında elde edilen değer int türünün sınırları içerisinde kaldığından doğrudan bu değer alınır. Örneğin:

#include <stdio.h>

int main()

{

double a = 50000000000.5673;

short b = (short) a;

printf("a = %lf\n",a);

printf("b = 0x%X\n",b);

printf("b = %d\n",b);

}

Burada dönüştürme byte türüne yapıldığından sayının noktadan sonraki kısmı atıldığında değer int türüne dönüştürülür. Sayının noktadan sonraki kısmı atıldığında elde edilen değer int türü sınırları dışındadır. Çok büyük bir değer olarak (pozitif olarak) sınırlar dışında kaldığından bu sayı int türü içerisinde tutulan en büyük sayıya (2147483647) dönüştürülür. Daha sonra sayının yüksek anlamlı 3(üç) byte’ı atılarak 1(bir) byte lık sayı elde edilir. Dikkat edilirse bu sayı -1 sayısıdır.