Selçuk Üniversitesi **ISSN 1302/6178** *Journal of Technical-Online*
Teknik Bilimler Meslek Yüksekokulu *Volume 8, Number:3-2009*
Teknik-Online Dergi
Cilt 8, Sayı:3-2009

BİLGİSAYAR SAYILARININ C PROGRAMLAMA DİLİNE UYGUN GENİŞLETİLMESİ*

Ali Osman ÇIBIKDİKEN[1] Kemal AYDIN[2]

[1] Selçuk Üniversitesi, Kadınhanı Faik İçil MYO, Bilgisayar Programcılığı, Konya

[2] Selçuk Üniversitesi, Fen Fakültesi, Matematik Bölümü, Konya

Özet

Günümüzde fen ve mühendislik alanındaki problemler sayısal olarak bilgisayarlarda çözülmektedir. Problemin çözümünde kullanılan klasik matematiksel yöntemler, bilgisayarda her zaman istenilen çözümü veremeyebilir. Bunun nedenlerinden birisi, bilgisayarların reel sayılar yerine sonlu kümelerle çalışmasıdır. Bilgisayar sayılarının kümesi (Format), yazılım ve donanım açısından standartlara bağlanmıştır. C dilinde reel sayılarla çalışmak için *float*, *double* ve *long double* olmak üzere 3 çeşit bilgisayar sayı kümesi kullanılmaktadır. Bu çalışmada C dilindeki bu 3 çeşit kümeye karşılık gelen Format kümeleri incelenmiştir. Fakat "ucuz" Formatın genişletilmesi, "reel sayıların" ayrık kuvvetle gösterilmesi yöntemiyle yeni Format'ları verir. Elde edilen bu Formatların karakteristikleri çalışmamızda tablolar şeklinde verilmiştir.

Anahtar kelimeler: Bilimsel hesaplama, bilgisayar sayıları, programlama dili.

* Bu çalışmadaki sonuçlar "Elemanter Matris İşlemlerinde Hata Tahmini" isimli yüksek lisans tezinden derlenmiştir.

Selçuk Üniversitesi **ISSN 1302/6178** *Journal of Technical-Online*
Teknik Bilimler Meslek Yüksekokulu *Volume 8, Number:3-2009*
Teknik-Online Dergi
Cilt 8, Sayı:3-2009

ON EXTEND OF COMPUTER NUMBERS IN C PROGRAMMING LANGUAGE

Abstract

Nowadays, problems in science and engineering are generally computed on computers by numerical methods. Sometimes the classical methods that are used on solving problems can not give the correct solutions. Because computers use finite computer numbers instead of real numbers. Computer numbers' types have standarts for hardware and software. C programming language can use 3 computer number types which are float, double and long double to work with real numbers. In this study, we examine this 3 number types in C programming language. But extending "cheap" Format obtains new Formats with using separate exponent method. These new Formats has been showed by tables.

Keywords: Scientific computing, computer numbers, programming language.

1. Giriş

Bilimsel hesaplamalar sırasında probleme ait veri girişi, işlemler ve elde edilen sonuçlar, bilgisayar sayıları kümesinin elemanları ile yapılmaktadır. Bilgisayar sayıları kümesi; sonlu büyüklükte, en büyük ve en küçük elemanı olan, elemanları arasında belli aralıkların bulunduğu sayılardan oluşan bir kümedir. Bilgisayar sayıları rasyonel sayıların sonlu bir alt kümesi ile çalışmaktadır.

(γ, p_- , p_+ , k) parametrelerine bağlı olarak,

$$\mathbf{F}^+ = \mathbf{F}^+ (\gamma, p_- , p_+ , k) = \{ \boldsymbol{u} \mid \boldsymbol{u} = \gamma^{p(\boldsymbol{u})} m(\boldsymbol{u}) \}$$

ve γ sayı sisteminin tabanını (genelde 2, 8, 10 veya 16 kullanılmaktadır) göstermek üzere,

$$\mathbf{F}= \{0\} \cup \mathbf{F}^+ \cup \mathbf{F}^- ; \quad \mathbf{F}^- = - \mathbf{F}^+ \tag{1}$$

şeklinde tanımlanan kümeye *bilgisayar sayıları kümesi* veya *Format kümesi* denilmektedir [1, 3, 7, 9, 10, 11]. Burada $p_- \in \mathbf{Z}^-$, k, $p_+ \in \mathbf{Z}^+$ olmak üzere, $p_- \le p(u) \le p_+$, $p(u) \in \mathbf{Z}$ ve

$$m(\boldsymbol{u}) = \frac{m_1}{\gamma} + \frac{m_2}{\gamma^2} + ... + \frac{m_k}{\gamma^k} ; \quad m_j \in \mathbf{Z},\ 0 \le m_j \le \gamma - 1,\ j = 1, 2, \ldots, k\ (m_1 \neq 0) \tag{2}$$

şeklinde tanımlıdır [1, 5, 9]. **F** kümesinin sonlu sayıda rasyonel sayılardan oluştuğu, yani **F** sonlu ve $\mathbf{F} \subset \mathbf{Q}$ olduğu açıktır. Ayrıca burada,

- $p(\boldsymbol{u})$: kuvveti
- $m(\boldsymbol{u})$: mantisi ($\frac{1}{\gamma} \le m(\boldsymbol{u}) < 1$)

göstermektedir. $q_- \le p_-$, $p_+ \le q_+$, k < s ise, $\mathbf{F}(\gamma, p_- , p_+ , k) \subset \mathbf{F}(\gamma, q_- , q_+ , s)$ dir. $\mathbf{F}(\gamma, p_- , p_+ , k)$ formatına daha ince format, $\mathbf{F}(\gamma, q_- , q_+ , s)$ formatına dahakaba format denir [1].

$p(u)$ değerinin negatif olmaması 0'a yakın sayılarla çalışmamızı engellemektedir. Çünkü 0'a yakın sayılarla çalışabilmek için kuvvet değerinin negatif olması gerekmektedir. Ayrı bir hafızada sakladığımız sayıyı $p(u)$ değerinden çıkartarak negatif değer elde edilebilir. Saklanılan bu sayı q ise $p(u)$; $p_- - q \le p(u) \le p_+ - q$ arasındaki tam sayıları alacaktır.

2. Bilgisayar Sayıları Kümesi Standartı

Format kümesinde $\gamma = 2$ olarak seçilirse, bilgisayarların yazılım ve donanımlarında standart oluşturan IEEE Standartı karşımıza gelir. IEEE Standartı *tek* (*single*) hassasiyet ve *çift* (*double*) hassasiyet olmak üzere iki kısımdan oluşmaktadır [2, 6]. IEEE Standartında bilgisayar sayıları Tablo 1 de gösterilmektedir. [2, 6, 11].

Tablo 1. IEEE Standartına göre bilgisayar sayılarının gösterimi

	Tek Hassasiyet	**Çift Hassasiyet**
Bilgisayar sayısı	$(-1)^s \times 1.m \times 2^{e-127}$	$(-1)^s \times 1.m \times 2^{e-1023}$

IEEE Standartının tek hassasiyet ve çift hassasiyet durumuna göre bit dağılımı Tablo 2. de gösterilmiştir [2].

Tablo 2. IEEE Standartına göre bitlerin dağılımı.

	Tek Hassasiyet	**Çift Hassasiyet**
s (işaret biti)	1 tane	1 tane
e (kuvvet biti)	8 tane	11 tane
m (mantis biti)	23 tane	53 tane

Mantisin ilk elemanı 0 dan farklı bir eleman olması gerektiği (2) de verilmişti. Buna göre IEEE standartında mantis için ayrılan bitin değeri 0 olamaz. Bit olarak işlem yapıldığına göre bu değer 0 değilse 1 olmalıdır. Mantis için ayrılan alandaki ilk eleman daima 1 olacağından, hafızada ayrıca yer kaplamaması için buna "gizli bit (*hidden bit*)" denilmektedir [2, 3, 6].

Bilgisayar yazılımları hazırlanırken reel sayıların kullanımı için değişken tiplerinin tanımlanması gerekmektedir. Böylece bilgisayarın kullanabileceği sayılar belirlenir. Sonlu sayıda olan bu sayılar, gerekli değişken tipine atanmaz ise sonuçlar hatalı çıkabilir. Bu aslında kullanılacak Format kümelerinin tanıtılmasından başka bir şey değildir. Tablo 3'de C programlama dilinde reel sayıların saklanması için kullanılabilecek değişken tipleri ve özellikleri verilmiştir [8]. Tablo 3'de *m* değerlerine gizli bit dahil edilmiştir.

Tablo 3. C programlama dilinde kullanılan değişken tipleri.

Değişken tipi	Bellek alanı	*q*	Bitlerin kullanımı		
			s	*e*	*m*
float	4 byte	127	1	8	24
double	8 byte	1023	1	11	53
long double	10 byte	16838	1	15	65

C programlama dilinde kullanılan değişken tipleri, format kümeleri olarak Tablo 4'de verilmiştir.

Tablo 4. C dilinde kullanılan değişkenlere ait format kümeleri.

Değişken tipi	**γ**	p_-	p_+	***k***	**Format kümeleri**
float	2	-128	127	24	$\mathbf{F}_f(2,-128,127,24)$
double	2	-1024	1023	53	$\mathbf{F}_d(2,-1024,1023,53)$
long double	2	-16384	16383	65	$\mathbf{F}_{lgd}(2,-16384,16383,65)$

Tablo 4'den de açıkça görüleceği gibi $\mathbf{F}_f \subset \mathbf{F}_d \subset \mathbf{F}_{lgd}$ dir.

3. C Programlama Dilinde Bilgisayar Sayı Kümelerinin Genişletilmesi

Bir format kümesinde sayının kuvvet değeri, onun için ayrılan alanın büyüklüğüne ve kaydırma değeri ile sınırlıdır. Bu sınırı genişletmek için "ayrık kuvvet" yöntemi yardımcı olmaktadır. Bu sebeple özet kısmında "ucuz" kelimesi ile standartlar kullanılarak, ek bir maliyet getirmeden ayrık kuvvet ile daha ince format kümeleri elde edilebileceği kastedilmiştir.

Tablo 5. C programlama dilinde tam sayı için kullanılan değişken tipleri.

Değişken tipi	**Bellek alanı**	**Alt sınır**	**Üst sınır**
unsigned char	1 bye	$q_{uc-} = 0$	$q_{uc+} = 255$
unsigned int	2 byte	$q_{us-} = 0$	$q_{us+} = 65535$
char signed char	1 byte	$q_{c-} = -128$	$q_{c+} = 127$
int short int signed int signed short int	2 byte	$q_{i-} = -32768$	$q_{i+} = 32767$
long long int signed long int	4 byte	$q_{l-} = -2147483648$	$q_{l+} = 2147483647$
unsigned long int	4 byte	$q_{ul-} = 0$	$q_{ul+} = 4294967295$

Ayrık kuvvet yöntemi, p_- ve p_+ sayılarının standart olarak verilenden daha büyük kullanarak genişletilmesiyle sağlanmaktadır. Bu genişletmede, p_- ve p_+ için tam sayı

saklanacak alanlar yeterli olacaktır. Tablo 5'de C programlama dilinde tam sayı saklamak için kullanılan değişkenler gösterilmiştir [8]. Böylece bu değişkenlerdeki değerler p_- ve p_+ için kullanılabilecektir. Tablo 5'te verilen, q_{c-}, q_{c+}, q_{s-}, q_{s+}, q_{l-}, q_{l+}, değerlerini kuvvet değerleri için, mantisa değerlerini standart değerler kullanmak üzere yeni format kümeleri Tablo 6'da tanıtılmıştır [4].

Tablo 6. C programlama dilinde genişletilerek elde edilen format kümeleri

Kuvvet / **Mantisa**	**float**	**double**	**long double**
char	$\mathbf{F}_{cf}$	$\mathbf{F}_{cd}$	$\mathbf{F}_{cl}$
int	$\mathbf{F}_{if}$	$\mathbf{F}_{id}$	$\mathbf{F}_{il}$
long	$\mathbf{F}_{lf}$	$\mathbf{F}_{ld}$	$\mathbf{F}_{ll}$

Böylece kuvvet değerleri genişletilerek C dilinde standart olarak verilmiş olan kümelerin dışında 9 farklı Format kümesi elde etmiş oluruz.

Bu kümeler;

$\mathbf{F}_{cf} = \mathbf{F}(2, q_{c-}, q_{c+}, 24) = \mathbf{F}(2, -128, 127, 24)$

$\mathbf{F}_{if} = \mathbf{F}(2, q_{s-}, q_{s+}, 24) = \mathbf{F}(2, -32768, 32767, 24)$

$\mathbf{F}_{lf} = \mathbf{F}(2, q_{l-}, q_{l+}, 24) = \mathbf{F}(2,-2147483648, 2147483647, 24)$

$\mathbf{F}_{cd} = \mathbf{F}(2, q_{c-}, q_{c+}, 53) = \mathbf{F}(2, -128, 127, 53)$

$\mathbf{F}_{id} = \mathbf{F}(2, q_{s-}, q_{s+}, 53) = \mathbf{F}(2, -32768, 32767, 53)$

$\mathbf{F}_{ld} = \mathbf{F}(2, q_{l-}, q_{l+}, 53) = \mathbf{F}(2, -2147483648, 2147483647, 53)$

$\mathbf{F}_{cl} = \mathbf{F}(2, q_{c-}, q_{c+}, 65) = \mathbf{F}(2, -128, 127, 65)$

$\mathbf{F}_{il} = \mathbf{F}(2, q_{s-}, q_{s+}, 65) = \mathbf{F}(2, -32768, 32767, 65)$

$\mathbf{F}_{ll} = \mathbf{F}(2, q_{l-}, q_{l+}, 65) = \mathbf{F}(2, -2147483648, 2147483647, 65)$

olarak bulunur [4]. Burada,

$$\mathbf{F}_{cf} = \mathbf{F}_{f}$$

$$\mathbf{F}_{cd} \subset \mathbf{F}_{d}$$

$$\mathbf{F}_{cl} \subset \mathbf{F}_{l}$$

olduğu görülmektedir. O halde $\mathbf{F}_{cf}$, $\mathbf{F}_{cd}$, $\mathbf{F}_{cl}$ format kümelerini elde etmek bize bir şey kazandırmaz. Dolayısıyla $\mathbf{F}_{if}$, $\mathbf{F}_{lf}$, $\mathbf{F}_{id}$, $\mathbf{F}_{ld}$, $\mathbf{F}_{il}$, $\mathbf{F}_{ll}$ kümelerinin format parametre değerleri Tablo 7'de gösterilmiştir.

Tablo 7. C programlama dilinde genişletilmiş $\mathbf{F}_{if}$, $\mathbf{F}_{lf}$, $\mathbf{F}_{id}$, $\mathbf{F}_{ld}$, $\mathbf{F}_{il}$, $\mathbf{F}_{ll}$ kümelerinin Format parametre değerleri.

Format kümesi	**γ**	p_-	p_+	k
$\mathbf{F}_{if}$	2	-32768	32767	24
$\mathbf{F}_{lf}$	2	-2147483648	2147483647	24
$\mathbf{F}_{id}$	2	-32768	32767	53
$\mathbf{F}_{ld}$	2	-2147483648	2147483647	53
$\mathbf{F}_{il}$	2	-32768	32767	65
$\mathbf{F}_{ll}$	2	-2147483648	2147483647	65

Tablo 7 den görüldüğü gibi,

$$\mathbf{F}_{if} \subset \mathbf{F}_{lf}$$

$$\mathbf{F}_{id} \subset \mathbf{F}_{ld}$$

$$\mathbf{F}_{il} \subset \mathbf{F}_{ll}$$

olduğu açıktır [4].

4. Sonuç

Bilgisayar sayıları kümesi, C programlama dilinde standarta bağlı kalınarak, bilgisayar kapasitesine göre genişletilmiştir. Bu genişletme ile elde edilen Tablo 7'deki

p- değerleri mutlak değerce büyüdükçe, 0'a yakın bilgisayar sayılarının artması sonucu bilgisayar sayıları kümesi daha da inceldiğinden, hesaplamaların yüksek hassasiyetle yapılabilmesine imkan sağlanmıştır.

Genişletmeye yönelik algoritmalar hesaplama algoritmasına eklenerek daha hassas hesaplamalar yapan yazılımlar üretmek mümkün olabilecektir.

Kaynaklar

[1] Akın Ö ve Bulgak H, Lineer fark denklemleri ve kararlılık teorisi, Selçuk Üniversitesi Uygulamalı Matematik Araştırma Merkezi Yayınları No.2, Konya, 1998.

[2] ANSI/IEEE, IEEE Standard for binary floating point arithmetic, Std 754–1985, New York, 1985.

[3] Behrooz P, Computer architecture: From microprocessors to supercomputers, Oxford University Press, 2005.

[4] Çıbıkdiken A.O., Elemanter matris işlemlerinde hata tahmini, Selçuk Üniversitesi Fen Bilimleri Enstitüsü Uygulamalı Matematik Anabilim Dalı Yüksek Lisans Tezi, Konya, 2002.

[5] Godunov S. K., Antonov A. G., Kiriluk O. P. and Kostin V. I., Guaranteed accuracy in mathematical computations, Englewood Cliffs, N. J., Prentice- Hall, 1993.

[6] Goldberg D., What every computer scientist should know about floating-point arithmetic, Association for Computing Machinery, 1991.

[7] Kahan W., Lecture notes on the status of IEEE standart 754 for binary floating point arithmetic, University of California, 1997.

[8] Kernighan B.W, Ritchie D.M, C Programming Language, 2nd Edition, Prentice Hall, Software Series, 1988.

[9] Kulisch U.W, Mathematical foundation of computer arithmetic, IEEE Transactions of Computers, Vol. C-26, No.7, 1977.

[10] Kulisch U.W and Miranker W.L., Computer arithmetic in theory and practice, Academi Press Inc., 1981.

[11] Overton M.L., Numerical computing with IEEE floating point arithmetic, SIAM, 2001.