Bits και Bytes:
από τον char στον double

ή

πως να διεκδικήσετε
- και να κερδίσετε -
35 σημαντικά ψηφία στους υπολογισμούς σας

:-)

by Ch Iossif @ 2010

Bits - Bytes

Το bit (μπιτ) είναι ένα δυαδικό ψηφίο, το οποίο μπορεί να πάρει τις τιμές 0 ή 1. Η συντομογραφία bit προέρχεται από τη σύντμηση των λέξεων BInary digiT.

Το byte (μπάιτ) είναι μονάδα μέτρησης ποσότητας πληροφορίας στα υπολογιστικά συστήματα. Ένα byte ισοδυναμεί με 8 bit.

Bits - Bytes

Πολλαπλάσια του byte είναι τα:

- Kilobyte (Κιλομπάιτ), 1 kB = 1.024 bytes = 2^10 bytes
- Megabyte (Μεγαμπάιτ), 1 MB = 1.048.576 bytes = 2^20 bytes
- Gigabyte (Γιγαμπάιτ), 1 GB = 1.073.741.824 bytes = 2^30 bytes
- Terabyte (Τεραμπάιτ), 1 TB = 1.099.511.627.776 bytes = 2^40 bytes
- Petabyte (Πεταμπάιτ), 1 PB = 1.125.899.906.842.624 bytes = 2^50 bytes
- Exabyte (Εξαμπάιτ), 1 EB = 1.152.921.504.606.846.976 bytes = 2^60 bytes
- Zettabyte (Ζεταμπάιτ), 1 ZB = 1.180.591.620.717.411.303.424 bytes = 2^70 bytes
- Yottabyte (Γιωταμπάιτ), 1 YB = 1.208.925.819.614.629.174.706.176 bytes = 2^80 bytes

Απλές δομές δεδομένων

Βαθμωτά μεγέθη:

- Χαρακτήρας
- Ακέραιος
- Απαρίθμηση
- Δείκτης

Μη βαθμωτά μεγέθη:

- Πραγματικοί αριθμοί
- (αριθμοί κινητής υποδιαστολής)

Απλές δομές δεδομένων στην C

size of char is 1

size of short is 2

size of int is 4

size of long is 8 (4)

size of long long is 8

size of float is 4

size of double is 8

size of long double is 16 (12)

σε bytes με 64 (32) bit αρχιτ.

Απλές δομές δεδομένων στην C

```
#include <stdio.h>
int main(void){
    printf("size of char is %u\n", sizeof(char));
    printf("size of short is %u\n", sizeof(short));
    printf("size of int is %u\n", sizeof(int));
    printf("size of long is %u\n", sizeof(long));
    printf("size of long long is %u\n\n", sizeof(long long));
    printf("size of float is %u\n", sizeof(float));
    printf("size of double is %u\n", sizeof(double));
    printf("size of long double is %u\n", sizeof(long double));
    return 0;
    }
```

Ακέραιοι και συμπλήρωμα ως προς 2

```
0 0 0 0 0 0 0 0 =   0
0 0 0 0 0 0 0 1 =   1
0 0 0 0 0 0 1 0 =   2
...
0 1 1 1 1 1 1 0 = 126
0 1 1 1 1 1 1 1 = 127
1 0 0 0 0 0 0 0 = 128   = -128
1 0 0 0 0 0 0 1 = 129   = -127
...
1 1 1 1 1 1 1 0 = 254   = -2
1 1 1 1 1 1 1 1 = 255   = -1
```

Πραγματικοί και σημαντικά ψηφία

Σημαντικά ψηφία × βάση ^ εκθέτης

πχ. σ.ψ. = 4, βάση=10, εκθέτης =1:

A= +1234.0---> +0.1234*10^+4 ---> +1234+4

B= +0.123 ---> +0.123x*10^+0 ---> +123χ+0

Γ=+1234.123 ---> +0.1234*10^+4 ---> +1234+4

άρα Γ=A+B και Γ=A με B ≠ 0 κι όμως ισχύει :-)

Πραγματικοί και σημαντικά ψηφία

Τύπος	μέγεθος	σ.ψ.<10>	σ.ψ.<2>
double:	8	16	53
long double:	16	20	64
__float128:	16	35	113

Πραγματικοί και σημαντικά ψηφία

```
#include <stdio.h>
void main(void) {
    char *p;
    int i,k;
    double da, db, dr;
    p=(char *)&da;

...
```

```
...
da=db=(double)1.0;
dr=da/(double)10; /* BASE */
da+=dr;
for (k=0;da>db;k++) {
    da=db;
    da+=dr;
    dr/=(double)BASE;
    }
printf("Double:\t\tsizeof=%lu\t\tsignificant digits=
    %d\n", sizeof(double), k);
}
```

Πραγματικοί σ.ψ. και γλώσσα μηχανής

```
movl    -88(%rbp), %edx
movq    %rax, -80(%rbp)
movl    %edx, -72(%rbp)
fldt  -80(%rbp)
fldt  -112(%rbp)
faddp    %st, %st(1)
fstpt-80(%rbp)
fldt  -112(%rbp)
fldt  .LC4(%rip)
fdivrp    %st, %st(1)
fstpt-112(%rbp)
addl $1, -24(%rbp)
```

```
movdqa .LC6(%rip), %xmm0
movdqa %xmm0, -144(%rbp)
movdqa -144(%rbp), %xmm0
movdqa %xmm0, -128(%rbp)
movdqa .LC7(%rip), %xmm1
movdqa -128(%rbp), %xmm0
call __divtf3
movdqa %xmm0, -160(%rbp)
movdqa -160(%rbp), %xmm1
movdqa -128(%rbp), %xmm0
call __addtf3
movdqa %xmm0, -128(%rbp)
```

Αναφορές

* Η wikipedia στα λήμματα
* * Two's complement
* * Floating point
* * IEEE 754-1985 και IEEE 754-2008
* http://www.x86-64.org/documentation/abi.pdf

και φυσικά:

η απληστία μου για περισσότερα σ.ψ.