

Programação e Desenvolvimento de Software I

Memoria

Prof. Héctor Azpúrua
(slides adaptados do Prof. Pedro Olmo)

Exibir o maior numero

- O que esse código esta realizando?
 - Exibir o maior número inteiro que pode ser representado no computador
- O que acontece quando o maior número é atingido e somamos UM?
 - Overflow de inteiro!
 - Ele “da a volta”
 - $255 + 1 = 0$
 - $255 + 2 = 1$

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
    char a;
    unsigned char b;

    a = pow(2, 7) - 1;
    b = pow(2, 8) - 1;
    printf("a = %d\t\tb = %d\n", a, b);

    a = a + 1;
    b = b + 1;
    printf("a = %d\t\tb = %d\n", a, b);

    return 0;
}
```

Qual o maior número inteiro?

- Para o compilador GCC, números inteiros são representados usando-se 32 bits (4 bytes)
- Como o bit mais significativo representa o sinal, sobram 31 bits para representar o valor do número (complemento-de-2). O maior inteiro será:

$$01111111111111111111111111111111 = 2^{31} - 1 = \mathbf{2147483647}$$

- Como assim? 😲
 - Com n bits, podemos representar 2^n números distintos
 - Sendo o maior número $2^n - 1$
 - Exemplo:
 - Para $n = 2$, temos 4 números possíveis, sendo 3 o maior número

Decimal	Binary
0	0
1	1
2	10
3	11

Complemento-de-2

- **Atenção!**

- Na representação em complemento-de-2 existe sempre um valor negativo a mais.

Binário	0000	0001	0010	0011	0100	0101	0110	0111
Decimal	0	+1	+2	+3	+4	+5	+6	+7
Binário	1000	1001	1010	1011	1100	1101	1110	1111
Decimal	-8	-7	-6	-5	-4	-3	-2	-1

Menor inteiro

- Assim, o menor valor inteiro representável:
 - Não será: -2147833647, mas sim -2147833648
- Como assim?
 - Com n bits, o menor número representável será -2^{n-1}
 - Exemplo: para $n = 4$, o menor número representável é $-2^3 = -8$
- Portanto, as variáveis do tipo **int** poderão armazenar valores no intervalo de:
 - -2147833648 a 2147833647

Modificadores de tipo

- A linguagem C define alguns **modificadores de tipo**. Alguns deles são:
 - **short**
 - **long**
 - **unsigned**
- Um modificador de tipo **altera o intervalo de valores** que uma variável pode armazenar
 - Ao tipo **float** não se aplica nenhum dos modificadores
 - Ao tipo **double** aplica-se apenas o modificador **long**
 - Ao tipo **char** aplica-se somente o tipo **unsigned**
- O modificador de tipo **short** instrui o compilador a representar valores inteiros usando 16 bits
 - Logo, uma variável **short int** pode armazenar valores inteiros no intervalo:
 - -2^{15} a $2^{15} - 1$

Modificadores de tipo

- Para as variáveis do tipo **char**, o compilador reserva 8 bits
- Assim, variáveis do tipo **char** podem armazenar valores inteiros no intervalo:
 - -2^7 a $2^7 - 1$
- O modificador de tipo **unsigned** instrui o compilador a não considerar o primeiro bit como sinal. Assim, variáveis **unsigned char** podem representar valores positivos maiores
 - O maior valor será: $2^8 - 1$

Exibir o maior numero

- No programa a direita, são atribuídos os maiores valores possíveis às variáveis a e b

```
a = 127          b = 255
a = -128         b = 0
x = 2147483647   y = 32767
x = -2147483648  y = -32768
```

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```

```
int main() {
    int x;
    short int y;
    char a;
    unsigned char b;
```

```
a = pow(2, 7) - 1;  
b = pow(2, 8) - 1;  
printf("a = %d\t\tb = %d\n", a, b);
```

```
a = a + 1;  
b = b + 1;  
printf("a = %d\t\tb = %d\n", a, b);
```

```
x = pow(2, 31) - 1; // maior int possivel
y = pow(2, 15) - 1; // maior short int possivel
printf("x = %d\tty = %d\n", x, y);
```

```
x = x + 1;  
y = y + 1;  
printf("x = %d\\t\\ty = %d\\n", x, y);
```

```
return 0;
```

1

Modificadores de tipo

- Em seguida, os valores das variáveis são incrementados de 1
- O que acontece então?
 - Ocorre um **extravasamento** (overflow)!
- **Exemplo:** considere a variável `a`

```
char a;  
a = pow(2, 7) - 1;
```

0111 1111 + 127

1000 0000 -128

```
a = pow(2, 7) - 1;  
b = pow(2, 8) - 1;  
printf("a = %d\t\tb = %d\n", a, b);
```

```
a = a + 1;  
b = b + 1;  
printf("a = %d\t\tb = %d\n", a, b);
```

- Mais detalhes sobre os modificadores de tipo podem ser vistos aqui:
 - https://en.wikipedia.org/wiki/C_data_types

Sistema hexadecimal

- O **Sistema Hexadecimal (base 16)** é o mais usado para representar endereços de memória
- **Grande poder de compactação:**
 - Consegue representar 1 byte com apenas 2 dígitos!
 - Ou seja, cada 4 bits são representados por um único algarismo hexadecimal
- Neste sistema são utilizados 16 algarismos:
 - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Sistema hexadecimal

- A tabela lista a correspondência entre os sistemas **binário, decimal e hexadecimal**
- **Se uma variável acessa o endereço abaixo, como ele é codificado em binário?**

0022FF74



0000 0000 0010 0010 1111 1111 0111 0100

Hexa	Decimal	Binário
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Sistema hexadecimal

Conversão entre sistemas de numeração

Hexa	Decimal	Binário
1	1	0001
4	4	0100
7	7	0111
A	10	1010
B	11	1011
F	15	1111

- Para converter um valor no sistema hexadecimal para o correspondente valor no sistema decimal e vice versa

- O que devo fazer?

- Exemplo:

- $(ABAFA)_{16} = (703226)_{10}$

Casa	Valor da Casa	Lógica	Cálculo	Valor Decimal
5	A	$10 * (16^4)$	$10 * 65536 =$	655 360
4	B	$11 * (16^3)$	$11 * 4096 =$	45 056
3	A	$10 * (16^2)$	$10 * 256 =$	2 560
2	F	$15 * (16^1)$	$15 * 16 =$	240
1	A	$10 * (16^0)$	$10 * 1 =$	10
Soma				703 226

- $(4711)_{10} = (1267)_{16}$

Número	Decimal	Base	Resultado	Inteiro	Ajuste do Resto	Resto
	4711	/ 16 =	294,4375	294	$0,4375 * 16 =$	7
	294	/ 16 =	18,375	18	$0,375 * 16 =$	6
	18	/ 16 =	1,125	1	$0,125 * 16 =$	2
	1	/ 16 =	0,0625	0	$0,0625 * 16 =$	1

Perguntas?

- E-mail:
 - hector@dcc.ufmg.br
- Material da disciplina:
 - <https://pedroolmo.github.io/teaching/pdsI.html>
- Github:
 - <https://github.com/h3ct0r>



Héctor Azpúrua
h3ct0r