

Algoritmos

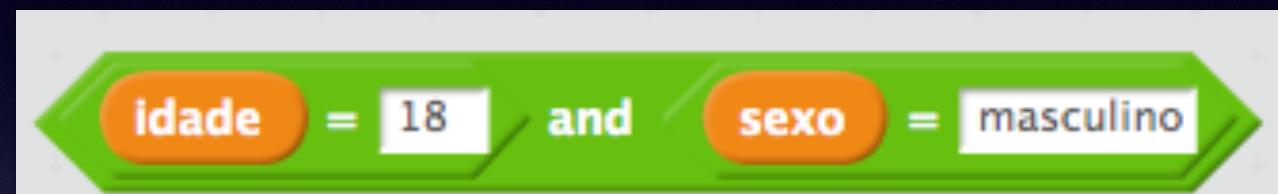
Introdução a Computação

Prof. Hitoshi Nagano, Ph.D.

Aula 5

Algebra Booleana

explique...



```
idade = 18 and sexo = masculino
```

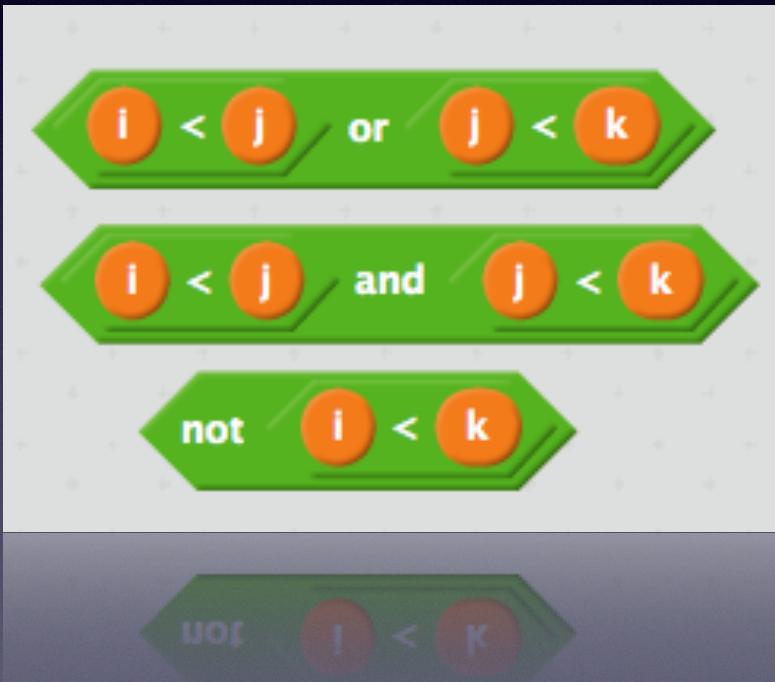


```
idade = 18 and sexo = feminino and voluntaria = sim
```



```
idade = 18 and sexo = masculino or idade = 18 and sexo = feminino and voluntaria = sim
```

expressões booleanas

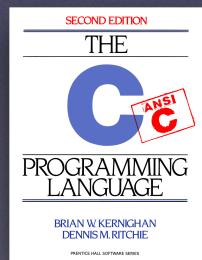


```
((i < j) || (j < k))  
((i < j) && (j < k))  
!(i < j)
```

expressões booleanas

Operador AND

condição A	condição B	resultado
0	0	0
0	1	0
1	0	0
1	1	1



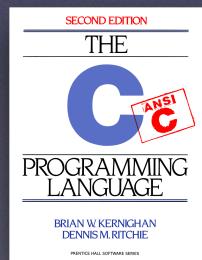
((condA) && (condB))



expressões booleanas

Operador OR

condição A	condição B	resultado
0	0	0
0	1	1
1	0	1
1	1	1



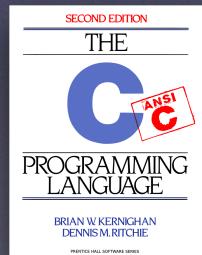
((condA) || (condB))



expressões booleanas

Operador NOT

condição	resultado
0	1
1	0



! (cond)



Exercício

- Escrever um programa para imprimir múltiplos de 3 ou 5
- Escrever um programa para imprimir múltiplos de 3 e 5

Loops aninhados

Nested Loops

Exercício a

- Escrever um programa para imprimir “#####” n vezes, um em cada linha.

- Exemplo: n = 2

#####

#####

- Exemplo: n = 5

#####

#####

#####

#####

#####

Exercício b

- Escrever um programa para imprimir “#” n vezes numa linha, em n linhas.
- Exemplo: $n = 2$
- Exemplo: $n = 5$

##

##

#####

#####

#####

#####

#####

Pergunta 1:
quantas linhas?

Exercício b

- Escrever um programa para imprimir “#” n vezes numa linha, em n linhas.
- Exemplo: $n = 2$
- Exemplo: $n = 5$

##

##

#####

#####

#####

#####

#####

Pergunta 2:
quantas “#” em cada linha?

Exercício c

- Escrever um programa para imprimir uma pirâmide.
- Exemplo: $n = 2$
- Exemplo: $n = 5$

##

Pergunta 1:
quantas linhas?

#####

Exercício c

- Escrever um programa para imprimir uma pirâmide.
- Exemplo: $n = 2$
- Exemplo: $n = 5$

```
#  
##
```

Pergunta 2:
quantas “#” em cada linha?

```
#  
##  
###  
####  
#####
```

Problemas de imprecisão e transbordamento

Tipos em C

- char 1 byte
- integer 4 bytes
- float 4 bytes
- bool 1 byte
- long 8 bytes
- double 8 bytes

Tipos em C

- char 1 byte
- **integer 4 bytes**
- **float 4 bytes**
- bool 1 byte
- long 8 bytes
- double 8 bytes

The screenshot shows a Microsoft Excel spreadsheet titled "Exemplo de sistema binário.xlsx". The formula bar displays the formula $=D3*D5$. The spreadsheet contains the following data:

	A	B	C	D	E	F	G	H	I	J
1										
2										
3		0	0	1	0	1	0	1	1	
4										
5		128	64	32	16	8	4	2	1	
6		0	0	32	0	8	0	2	1	
7										
8										
9		43								
10										
11										
12										
13										
14										
15										
16										
17										

The formula $=D3*D5$ is entered in cell D6. The result of the multiplication, 32, is displayed in cell E6. The binary representation of 32 is shown in row 6, where the 5th column from the left contains a 1 (representing 2^5) and all other columns contain 0s.

Pergunta

Quantos números diferentes consigo representar com 4 bytes?

Resposta: 4.294.967.296

unsigned int

inteiros positivos ($0, 2^{32} - 1$)

inteiros positivos e negativos ($-2^{31}, 2^{31} - 1$)

signed int

Genericamente

Quantos números diferentes consigo representar com n bits?

Resposta: 2^n

unsigned int

inteiros positivos ($0, 2^n - 1$)

inteiros positivos e negativos ($-2^{n-1}, 2^{n-1} - 1$)

signed int

Valores limites do Linux/C

(arquivo limits.h)

- 32-bits, podemos representar:
inteiros positivos ($0, 2^{32} - 1$)
inteiros positivos e negativos ($-2^{31}, 2^{31} - 1$)
- Em C no Linux, os limites operacionais de cada tipo encontram-se no arquivo: /usr/include/limits.h.

Veja, por exemplo, as linhas:

```
/* Minimum and maximum values a `signed int' can hold. */
#define INT_MIN      (-INT_MAX - 1)
#define INT_MAX      2147483647
/* Maximum value an `unsigned int' can hold. (Minimum is 0.) */
#define UINT_MAX     4294967295U
```

Valores limites do Linux/C

(arquivo limits.h)

De fato, no programa:

```
#include<stdio.h>
int main()
{
    unsigned int i = 4294967295;
    printf("%u\n",i);
    i++;
    printf("%u\n",i);
    i++;
    printf("%u\n",i);
}
```

Teremos como resultado:

```
~/Unisantos/prova$ gcc ex10.c -o
ex10
~/Unisantos/prova$ ./ex10
4294967295
0
1
```

maior inteiro que uma
variável tipo **int**
consegue representar

Deveria ser **4.294.967.296**
mas esse número não
consegue ser representado
com 32 bits

Inteiros

In the early days of computing, designers made computers express numbers using **unsigned binary**.

And they were content...

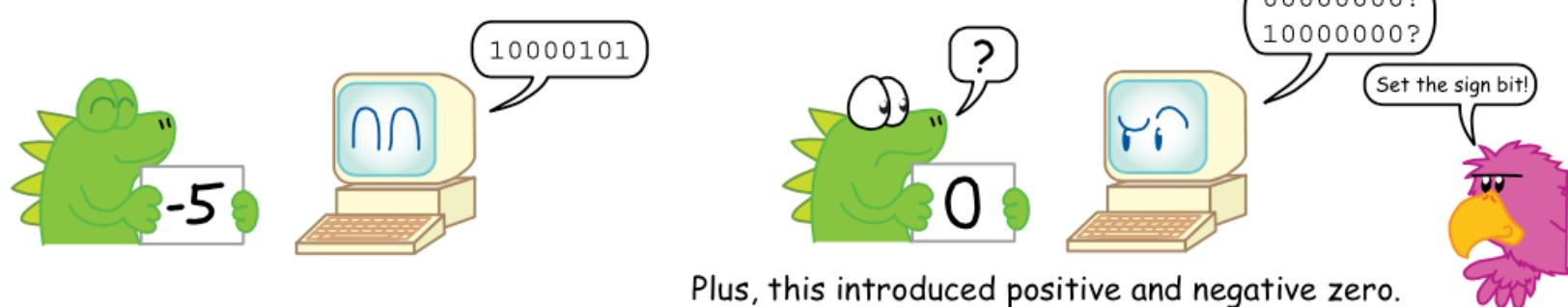
Until there were negative numbers.



To include negative numbers, designers came up with **sign magnitude**.

That took care of the negative numbers...

But the computer had to count backwards
for the negative numbers.



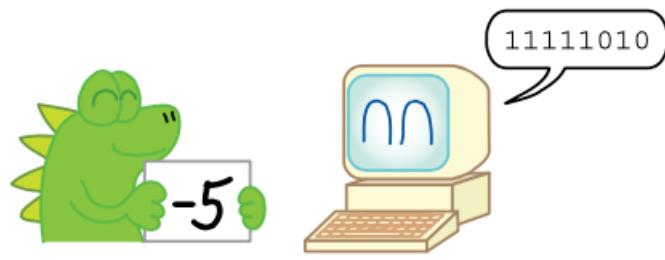
Plus, this introduced positive and negative zero.

Arte: Katrina Yim

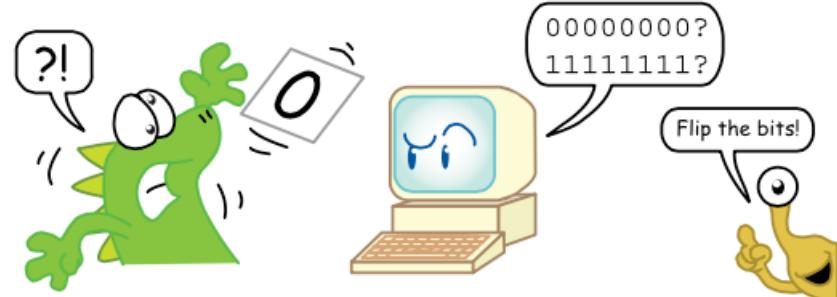
Inteiros

Then designers created **one's complement**.

Now computers only had to count
in one direction...

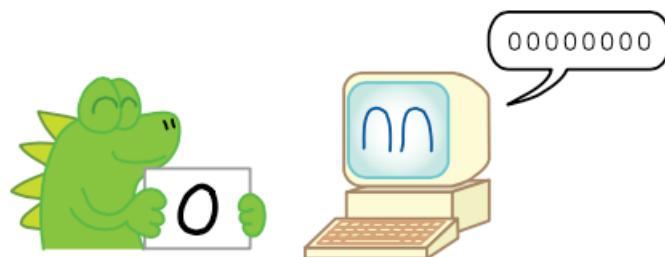


But there were still two zeroes!



Finally, designers developed **two's complement**.

Now, there was only one zero...



And they were content.



Katrina Yim

KatrinaYim

Inteiros

Positivo => Negativo

inverter os bits e somar 1

Flip the bits and
add 1!



Exemplo com 4 bits

Decimal	Binário			
-8	1	0	0	0
-7	1	0	0	1
-6	1	0	1	0
-5	1	0	1	1
-4	1	1	0	0
-3	1	1	0	1
-2	1	1	1	0
-1	1	1	1	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

Pergunta

como representar:

1/3

3.1415

1.08 x 10⁹

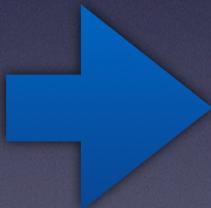
float



e: expoente (8 bits)

m: mantissa (23 bits)

s: sinal (1 bit)



valor = m.2^e

imprecisão do float

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

```
int main()
```

```
{
```

```
float f = 1/10;  
printf ("%f\n",f);
```

```
}
```

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

```
int main()
```

```
{
```

```
float f = 1.0/10;  
printf ("% .15f\n",f);
```

```
}
```

exemplos

0 =	0 = 0 00000000 00000000000000000000000000000000
-0 =	-0 = 1 00000000 00000000000000000000000000000000
0.125 =	0.125 = 0 01111100 00000000000000000000000000000000
0.25 =	0.25 = 0 01111101 00000000000000000000000000000000
2 =	2 = 0 10000000 00000000000000000000000000000000
4 =	4 = 0 10000001 00000000000000000000000000000000
0.375 =	0.375 = 0 01111101 10000000000000000000000000000000
3 =	3 = 0 10000000 10000000000000000000000000000000
6 =	6 = 0 10000001 10000000000000000000000000000000
0.1 =	0.1000000149011612 = 0 01111011 1001100110011001101
0.2 =	0.2000000298023224 = 0 01111100 1001100110011001101
0.8 =	0.80000001192092896 = 0 01111110 1001100110011001101
1e+12 =	999999995904 = 0 10100110 11010001101010010100101
inf =	inf = 0 11111111 00000000000000000000000000000000
-inf =	-inf = 1 11111111 00000000000000000000000000000000
nan =	nan = 0 11111111 10000000000000000000000000000000

Problemas

- transbordamento de inteiro
integer overflow
- imprecisão do ponto flutuante
floating point imprecision

Tipos em C

- char 1 byte
- integer 4 bytes → • long 8 bytes
- float 4 bytes → • double 8 bytes
- bool 1 byte

Imprecisão e suas consequências

- Arianne 5 e Patriot
<https://www.youtube.com/watch?v=EMVBLg2MrLs>

Paradoxo Simpson

Carla

Luiz

R\$100/R\$500

R\$200/R\$800

R\$600/R\$800

R\$400/R\$500

Carla

R\$100/R\$500

20%

75%

R\$600/R\$800

Luiz

R\$200/R\$800

25%

80%

R\$400/R\$500

Carla

R\$100/R\$500

20%



Luiz

R\$200/R\$800

25%

75%



80%

R\$600/R\$800

R\$400/R\$500

QUEM POUPA MAIS ???

Carla



Luiz



Carla



Luiz



Carla



Luiz



Carla



20%



Luiz



25%



75%



80%

Carla



Luiz



Carla



Luiz



Paradoxo de Simpson

Carla



Luiz

