Lista de exercícios 1

Tipos de Dados

Aluno: Gabriel Alessi Posonski RA: 2259583

1.

a) char: 8 bits, 256 números, {-127, 128}

b) unsigned char: 8 bits, 256 números, {0, 255}

c) short int: 16 bits, 65.536 números, {-32.768, 32.767}

d) unsigned short int: 16 bits, 65.536 números, {0, 65.535}

e) int: 32 bits, 4.294.967.296 números, {-2.147.483.648, 2.147.483.647}

f) unsigned int: 32 bits, 4.294.967.296 números, {0, 4.294.967.295}

g) (obs.: em sistemas de 32 bits, long int possui as mesmas características que um tipo 'int', os valores aqui apresentados serão levados em consideração um sistema de 64 bits.)
long int: 64 bits, 18.446.744.073.709.551.616 números, { -9.223.372.036.854.775.808, 9.223.372.036.854.775.807}

h) unsigned long int: 64 bits, 18.446.744.073.709.551.616 números, {0, 18.446.744.073.709.551.615}

i) double: 64 bits, 18.446.744.073.709.551.616 números, {10^ (-308), 10^ (308)}

2. R= 4) log _(2)32= 5 (log de 32 na base 2), onde 32 é a quantidade N de dados necessários para serem armazenados. Por se tratar de um sistema binário, a base da função é 2. Portanto seriam necessários no mínimo 5 bits para representar todas as opções.

3) 6 bits

4) Para cada pixel, seriam necessários 3 bits (2^3 = 8), portanto precisariam de 3N bits, onde N é a quantidade de pixels da tela para determinada imagem. Digamos que há uma imagem de 800 pixels, precisaríamos de 2400 bits na memória para armazená-la, o que dá 300 bytes de espaço necessário na RAM.

5)

a) 2.147.483.647 gramas

b) Seria válido avaliar se há a necessidade de tanta precisão no armazenamento do peso dos alunos. Supondo que seriam necessários apenas 2 dígitos de precisão a serem armazenados, o programador poderia utilizar o tipo 'unsigned short int', dessa maneira teríamos o intervalo de peso {0, 65.535}, o menor peso seria 0 e o maior de 655,35 kg. Como o tipo 'short int' ocupa somente metade dos bytes necessários para um tipo 'int', a economia no armazenamento do banco de dados seria dada por 16N, onde N é o número de alunos e teríamos o resultado em bytes. Ex: supondo que a academia tenha 600 alunos matriculados, a economia seria de 9600 bytes.

6) Antigamente o valor utilizado era int, 32 bits, e suportava até 2.147.483.647 visualizações (2^32, pois se trata de um sistema binário que necessita de 32 bits para armazenar tal dado), com intervalo entre -2.147.483.648 e 2.147.483.647. Com isso, perceberam que o valor era pequeno para o uso e passaram a utilizar o tipo 'unsigned long int', assim o sistema utilizaria o dobro de bits (2^64) e todo o intervalo trabalharia apenas com valores positivos, já que não faz sentido um contador de views ser negativo. O novo contador vai de 0 a 18.446.744.073.709.551.615 visualizações.

5) Esse é um dos trechos do relatório da ESA sobre o acidente:

The intemal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This resulted in an Operand Error. The data conversion instructions (in Ada code) were not protected from causing an Opemnd Error, although Other conversions of comparable variables in the same place in the code were protected.

Fonte: https://www.esa.int/Newsroom/Press_Releases/Ariane_501_-_Presentation_of_Inquiry_Board_report

O código em Ada acabou convertendo um número do tipo float (64 bits) para short int (16 bits), o número era maior que 32.767, então a conversão falhou.

L_M_BV_32: = TBD.T_ENTIER_32S ((1.0/C_M_LSB_BV) *G_M_INFO_DERIVE (T_ALG.E_BV));*
if L_M_BV_32> 32767

then P_M_DERIVE (T_ALG.E_BV): = 16 # 7FFF #;

elsif L_M_BV_32 <-32768

then P_M_DERIVE (T_ALG.E_BV): = 16 # 8000 #;

else P_M_DERIVE (T_ALG.E_BV): = UC_16S_EN_16NS (TDB.T_ENTIER_16S (L_M_BV_32));

end if;

P_M_DERIVE (T_ALG.E_BH): = UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH) G_M_INFO_DERIVE (T_ALG.E_BH)));

Trecho de código encontrado em: https://livreeaberto.com/um-erro-de-ponto-flutuante-que-causou-prejuizo-meio-bilhao

Em um código, a solução para o erro seria armazenar a conversão para um tipo float, que seria o mesmo tipo do número origem e assim evitaria um overflow no sistema.