Exercício T11 - Parte dois

Refaça, agora com o computador, alguns itens do T11

Tente executar em mais de um compilador se você tiver esta possibilidade. Dev++, CodeBocks, Linux, 32 ou 64 bits.

Nomes: Rafael Nossal

1) Qual o resultado das seguintes operações (faça sem computador. Depois verifique no compilador. Considere variáveis do tipo int)

a) `a=4;  a = ++a + ++a;` Valor do a: 12
b) `a=5;  a = ++a + a++;` Valor do a: 13
c) `a=33;  a = a-- + --a;` Valor do a: 64
d) `a=30;  a = --a + a++;` Valor do a: 59
e) `x=x^y;  y = x^y; x = x^y;` Valor de x e de y: x é o valor inicial de y e y o valor inicial de x

- Bateu com os valores que você calculou em aula?
- Houve alguma diferença entre compiladores? Se as variáveis forem do tipo char, mudou alguma coisa? Comente.

- Os compiladores CL e GCC obtiveram os mesmos resultados
- Se as variáveis forem char as questões A, B e D alteram os valores (11, 12 e 58 respectivamente). Aparentemente o compilador usando char segue as operações em ordem, não alterando o valor anterior já avaliado, como no exercício A-1 em que com inteiro por conta dos incrementos acaba sendo "6+6" (como se a primeira concatenação fosse "referenciada" da segunda) e com char acaba sendo "5+6".

2) Qual o valor de r após as expressões abaixo (elas são executadas em sequência, ou seja, cada uma pode influenciar as posteriores. Não use o compilador)

```
char a, b, c, d, e, r;
a = 13; b = 30; c = 2; d = 0; e = 3;
```

a) `r = a | b;` r = 31
b) `r = c-- | (~d);` r = -1
c) `r = a & b;` r = 12
d) `r = --a - --c;` r = 11
e) `r = (b >> --e) | b;` r = 31
f) `r = (a + c--) << (1499 / 500);` r = 60
g) `r = (b / e) * 2;` r = 20
h) `r = ( (a^b) * (c--|d) + 1 + e ) << 2;` r = -88
i) `r = ~c+1;` r = -2
j) `r = (a > b) == d;` r = 1 `/* Desafio. O que será que ocorre? */`

- Bateu com os valores que você calculou em aula? Se não, o que errou?
- Muda as respostas se as variáveis forem do tipo int? Porque?

- As respostas bateram tirando a questão letra H em que a expressão ficou complexa demais e ao final acebei por usar o compilador
- Não muda (exceto a letra H), pois as comparações em sua maioria são (tirando a letra D) são operações binárias e de forma binária eles são os mesmos.

3) Execute em seu compilador a seguinte sequencia

A) `int a; a = 200; a = (a*0.7)+a;`

Qual deveria ser o valor final do a? Qual foi o valor que o teu compilador gerou?

- O valor final deveria ser 340 e o compilador gerou 340 (compilador em uma máquina 64bits)

B) Quantas vezes o laço **deveria** ser executado? Quantas realmente executou? Procure testar em mais de uma versão de compilador. Mudou alguma coisa trocando a variável para double? Se trocar a comparação para 0.6 mudou algo?

```
float k;
for (k = 0.01; k!=0.7; k=k+0.01) {
     printf("k = %f\n",k);
}
```

```
float k;
for (k = 0.1; k!=0.7; k=k+0.1) {
     printf("k = %f\n",k);
}
```

- O laço deveria ser executado 69 vezes, mas executou infinitamente. Trocando para double o mesmo comportamento ainda ocorre e trocando para 0.6 o comportamento ainda se mantém.