Folha 1 das Aulas Práticas

Resolução

Programação Imperativa e Programação de Computadores
2004/2005

1. *Está correcta?*

   (a) `if(i==2) i++`
   Sintacticamente incorrecta: as instruções terminam com '**;**';
   Versão correcta:

   ```
   if(i==2) i++;
   ```

   Efeito: *se i é igual a dois, então incrementa i*;

   (b) `if(i=2) i++;`
   Sintacticamente correcta.
   Efeito[1]: *i toma o valor dois; se i não é zero então incrementa i*;
   Nota: em C, os testes lógicos podem ser constituídos por valores; nestes casos, o teste efectuado é ser diferente de zero, por exemplo:

   ```
   if(a) a++;     <=>     if(a!=0) a++;
   ```

   (c) `while(i>0 || i<=0) i+=2;`
   Sintacticamente correcta.
   Efeito: *enquanto i for maior do que zero ou menor ou igual a zero (condição universal), incrementa i em duas unidades*;
   Nota: esta instrução produz um ciclo infinito;

   (d) `while(i>0); i+=2;`
   Sintacticamente correcta.
   Efeito: *se i é maior do que zero, produz-se um ciclo infinito que em cada iteração não executa qualquer instrução; se i não é maior do que zero então incrementa i em duas unidades*;

   (e) `for(i=10;i>=0;i--) printf(¨*¨);`
   Sintacticamente correcta.
   Efeito: *i é inicializado com o valor dez; enquanto i for maior ou igual a zero, imprime * e decrementa i*;
   Nota: no final terá impresso ********** (onze iterações);

[1] provavelmente, o efeito pretendido seria o da alínea anterior, que pode ser obtido usando a referida sintaxe; notar a diferença entre atribuições (=) e comparações (==);

(f) `if i==10 printf(¨*¨);`
Sintacticamente incorrecta: a condição da instrução if tem de estar entre parêntesis;
Versão correcta:

```
if (i==10) printf(¨*¨);
```

Efeito: *se i é igual a dez então imprime **;

(g) `if(i==10) then printf(¨*¨)`
Sintacticamente incorrecta: em C não existe a palavra-chave **then**, e as instruções terminam com '**;**';
Versão correcta:

```
if (i==10) printf(¨*¨);
```

Efeito: *se i é igual a dez então imprime **;

2. *O que faz?*

(a)

```
s = 0; /* 's' é inicializado com 0 */
while(n>0){ /* enquanto 'n' for maior do que 0 executa: */
  s+=n; /* soma 's' e 'n' e atribui o resultado a 's' */
  n--; /* decrementa 'n' */
}
```

Valor final: se $n = 0$ então $s = 0$; se $n > 0$ então $s = \sum_{i=1}^{n} i$;

(b)

```
s = 0; /* 's' é inicializado com 0 */
while(n>=0){ /* enquanto 'n' for maior ou igual a 0 executa: */
  s=s+n; /* soma 's' e 'n' e atribui o resultado a 's' */
  n=n-1; /* decrementa 'n' */
}
```

Valor final: $\forall n \geq 0, s = 0 + ... + n = \sum_{i=0}^{n} i$

(c)

```
s = 0; /* 's' e 'i' são inicializados com 0 */
for(i=0;i<n;i++) /* enquanto 'i' for menor que 'n' executa: */
  s=s+i; /* soma 's' e 'i' e atribui o resultado a 's' */
         /* incrementa 'i' no fim de cada iteração */
```

Valor final: se $n = 0$ então $s = 0$; se $n > 0$ então $s = \sum_{i=0}^{n-1} i$;

(d)
```
for(s=0,i=0;i<n;i++) /* 's' e 'i' são inicializados com 0 */
  s=s+i; /* enquanto 'i' for menor que 'n' executa: */
         /* soma 's' e 'i' e atribui o resultado a 's' */
         /* incrementa 'i' no fim de cada iteração */
```

Valor final: se $n = 0$ então $s = 0$; se $n > 0$ então $s = \sum_{i=0}^{n-1} i$;

(e)
```
s = 0;  /* 's' e 'i' são inicializados com 0 */
for(i=0;i<n;i++) /* enquanto 'i' for menor que 'n' executa: */
  s=s+1; /* incrementa 's' */
         /* incrementa 'i' no fim de cada iteração */
```

Valor final: se $n = 0$ então $s = 0$; se $n > 0$ então $s = \sum_{i=0}^{n-1} 1 = n \times 1 = n$;

(f)
```
s = 0; /* 's' e 'i' são inicializados com 0 */
for(i=0;i<n;i++){ /* enquanto 'i' for menor que 'n' executa: */
  s=s+1; /* incrementa 's' */
  s=s-1; /* decrementa 's' */
  printf(¨s = %2d\n¨,s); /* imprime inteiro 's' */
}
```

Valor final: $\forall n \geq 0, s = 0$