DEPARTAMENTO DE FORMAÇÃO DE ENGENHEIROS NAVAIS - RAMO DE ARMAS E ELECTRÓNICA

3115 – SISTEMAS OPERATIVOS, ALGORITMOS, E ESTRUTURAS DE DADOS
4º ANO AEL
1ª Repetição Escrita de 2005/2006

Leia atentamente o enunciado. Seja breve nas respostas, mas justifique-as convenientemente. Por favor, use letra legível! Quando apresentar programas comente-os devidamente. Bom trabalho nesta repetição !

I

I.a) O programa seguinte tem algum erro de sintaxe ? Se sim corrija-o.
I.b) O que é que o programa escreve no ecrã quando executa ?

```
#incude <stdio.h>
#define TOT 5
extern int volta(int a, int *b );
int c:=3;
Main()
{
 int a=2;
 printf("%d %d\n",a,c);
 a=volta(TOT, &a)
 printf("%d %d\n",a,c);
 return 1;
}
int volta(int a, int *b )
{
 int d=7;
 printf("-%d %d %d %d\n",a,*b,c,d);
 if( a>1 )
   c=volta( a-3, &a);
 printf("-%d %d %d %d\n",a,*b,c,d);
 return (*b-1);
}
```

II

Defina uma estrutura de dados e escreva as rotinas necessárias para implementar pilhas ("stacks"). Cada elemento da pilha deve ser uma estrutura, que necessariamente tem que ter um campo "nome" com 30 caracteres, e 4 campos com inteiros "Track_no", "Dist", "Marc", e "Pri"). Deverá ser possível criar várias pilhas, e para qualquer deles deve ser possível chamar as seguintes rotinas:

ST_Create - Deve criar um stack, devolvendo um apontador para o novo stack.
ST_Push - Deve pôr um novo elemento no stack, e devolver como valor de roturno o número de elementos do stack (depois de acrescentado o elemento posto).
ST_Pop – Deve retirar um elemento do stack, e devolver como valor de roturno o número de elementos do stack (depois de retirado esse elemento).
ST_Size - Deve devolver como valor de roturno o número de elementos do stack.
ST_Clean – Deve apagar todos os elementos do stack, e apagar o stack.

III

Se não está a trabalhar para o UAV (Unmanned Aerial Vehicle) que está a desenvolvido aqui na Escola Naval, conhece bem quem esteja. Vamos nesta repetição escrita desenvolver algum do software necessário para esse projecto (software esse que até é bastante parecido com aquele que já desenvolveu para o PCROBOTS).
Uma das funções mais básicas consiste em ser capaz de percorrer um conjunto de "waypoints" previamente definidos. Para isso, o UAV dispõe de um GPS que indica a posição em cada ponto, e um sistema que permite apontá-lo em qualquer direcção. Para simplificar, vamos usar como coordenadas geográficas coordenadas de grid cartesianas x,y, em "unidades de grid" inteiras.
Para tornar o nosso sistema muito flexível, vamos guardar os waypoints que queremos percorrer numa lista ligada, usando apontadores. Cada elemento dessa lista terá a seguinte estrutura:

```
typedef stuct x{
        int     n;          /* Número do waypoint */
        long    x,y;        /* Coordenadas do waypoint */
        int     tolerance;  /* Erro máximo admissível para a aproximação a (x,y) */
        struct x *next;     /* apontador para o próximo waypoint  */
        }waypoint;
```

As rotinas de interface com o resto do sistema têm os seguintes protótipos:

```
void Get_Position(long &x, long &y);  /* Devolve em x,y as coordenadas de grid actuais*/
void Set_Direction(int dir);          /* Aponta o UAV na direção "dir"*/
```

I.a) Escreva uma rotina que recebe como parâmetro de entrada um apontador para um waypoint, e aponta o UAV na direção desse waypoint. Devido a perturbações de vento e às imperfeições do próprio sistema, é preciso ir corrigindo a direcção até que o UAV chegue ao waypoint (com um erro inferior à tolerância indicada no waypoint). Assume-se que o UAV voa a uma velocidade constante.

II.b) Escreva uma rotina que, dado um apontador para waypoint chamado "primeira", e um número "n", encontre o waypoint "n", retire esse waypoint da lista, e devolva o apontador para esse waypoint.

Bom trabalho !