Universidade do Minho

Transcrição de aula

Disciplina	Programação Imperativa – 1º ano - LEI		
Secretário	Número: 61065	Nome: Fábio Gomes	35 (alunos)
Data: 2011-04-26	TP3		Nº Página 5

SUMÁRIO

Resolução de exercícios com listas ligadas: Parques de Estacionamento e Grupos de Alunos.

```
typedef char Nome[10];
typedef struct sLugar
  {int lugar;
   struct sLugar * seg;
  } *Lugar, NLugar;

typedef struct sParque
  {int nLugares;
   Lugar ocupado;
   Nome nparque;
  } Parque;

typedef struct sLParque
  {Parque p;
   struct sLparque *seg;
  } *LParque, NLParque;
```

Transcrição (folha de continuação)

```
int main ()
{ Parque  p1 = {600, NULL, "P1"};
          p2 = {1500, NULL, "P2"};
  LParque cp=NULL;
  ...
  cp= inserirParque (cp,p1);
  cp= inserirParque (cp,p2);
  ...
  listarParques (cp);
  if (disponivel (cp,"P1",23))
     cp = estaciona (cp,"P1",23);
  ...
  listar disponibilidades (cp);

void listar (LParque lp)
  {if (lp)
     { printf ("%s, %d \n", lp->lg p. nparque, lp -> p. nlugares);
       listar ( lp -> seg )
     }
  }

void listardisponibilidade (Lparque lp)
 { if (lp)
     {printf ("%s %d \n", lp->p. nparque, count (lp -> p. nocupado));
      listardisponibilidade (lp -> seg);
     }
 }
```

lp -> p. nparque
ou
*lp . p . nparque

Transcrição (folha de continuação)

```
int disponivel (LParque cp , char * p , int nlugar)
{int res=0;
  while ( cp && res ==0)
 { if (strcmp (p , cp -> p . nparque )==0)
    { while (cp -> p. ocupados && res ==0)
      { if ( cp ->p. ocupados ->lugar == nlugar)
            res=1
     cp ->p . ocupado - cp -> p . ocupado -> seg ;
    }
  }
   cp = cp -> seg;
  }
  return res;
  }

LParque estacionar (LParque lp , char * nome , int lugar)
 { int flag =0 ;LParque aux = lp;
    while ( aux && ! flag )
     { if ( !strcmp (aux ->p . nparque , nome ))
     aux .ocupados =push (aux -> p . ocupados , nlugar);
          flag =1;
      }
     aux = aux -> seg;
      }
    return cp;
   }
```

Transcrição (folha de continuação)

```
LParque estaciona (LParque e, char *nome, int *lug)
{ Parque aux;
  if(e)
    if (!strcmp (e->p.nparque, nome))
      {aux = (Lugar) malloc (sizeof (Lugar))
       aux -> lugar = lug;
       aux -> seg = e -> p. ocupados;
       e -> ocupados = aux
       return e;
      }
    else
      { e -> seg = estaciona (e->seg, nome, e.lugar);
        return e;
      }
  }
```

Lista Grupos

Grupo – Lista Alunos
– Lista Notas

Aluno – nome
– número

```
typedef struct sAluno
  { char *nome;
    char *numero;
  } Aluno;

typedef struct sLNotas
  { float Nota;
    struct sLNota *seg;
  } *LNotas, NNotas;

typedef struct sLAluno
  { Aluno a;
    struct sLAluno *seg;
  } *LAluno, NAluno;

typedef struct sGrupo
  { LAluno la;
    ~~Notas Nota~~ c Nota ln;
    struct sGrupo *seg;
  } *Grupo, NGrupo;
```

Transcrição (folha de continuação)

```
Void listarAlunos (Grupo g)
{ if (g)
  { lista e (g -> la);
    listaAlunos (g -> seg);
  }
}
lista e (Laluno l)
{ if (l)
  { printf ("%s, %s \n", l -> a.nome, l -> a.numero);
    listae (l -> seg);
  }
}
```