Programação Estruturada

Folha de exercícios

2000/1001

O jogo dos diferentes

O "jogo dos diferentes" foi explicado nas aulas teóricas; as regras são as seguintes:

1. Inicialmente há um só monte, com n peças.
2. O utilizador e o computador alternam, dividindo um dos montes em 2 montes diferentes e não nulos (os montes de 1 e 2 pedras tornam-se assim inúteis!).
3. Perde o jogo o jogador que não tiver nenhuma jogada legal.

Exemplo de uso

```
Tamanho do monte inicial? 7
  [ 7 ]
Escolha o tamanho de um monte a dividir: 7
Qual o tamanho de uma das divisões? 3
  [ 3  4 ]          (fica assim depois da sua jogada)
  [ 1  4  2 ]       (fica assim depois da minha jogada)
Escolha o tamanho de um monte a dividir: 4
Qual o tamanho de uma das divisões? 1
  [ 1  1  2  3 ]    (fica assim depois da sua jogada)
Desculpe, ganhei!
  [ 1  1  2  1  2 ] (fica assim depois da minha jogada)
```

Exercícios

1. Mostre que, qualquer que seja o tamanho n do monte inicial, o jogo termina.
2. Teste o programa dado, determinando para os primeiros valores de n se é possível ganhar sempre o jogo que se inicia com um monte de n peças.
3. Modifique o programa de modo a o tornar "fool-proof", isto é, resistente a utilizações e valores de dados não previstos.
4. Explique o funcionamento do programa; considere em especial a função crucial, "`joga`" e a função "`computador`".

Listagem

```
//-----------------------------------------------------------------------
//  O JOGO DOS DIFERENTES
//-----------------------------------------------------------------------

#define MAXM 100

 int monte[MAXM];
 int m;

 main(){
  int fim=0, ele=1;
  void mostra(void);
  void utente(void);
  int  bloqueado(void);
  void computador(void);
  printf("Tamanho do monte inicial? ");
  scanf("%d",&m);
  monte[0]=m;
  m=1;

  while(!fim){
    mostra();
    if(ele){
      utente();
      fim=bloqueado();
      if(fim)
 printf("Parabens, ganhou\n");
    }
    else{
      computador();
      fim=bloqueado();
      if(fim)
 printf("Desculpe, ganhei!\n");
    }
    ele=(ele?0:1);
  }
  mostra();
}
```

```c
// Imprime o tamanho de cada monte ("o tabuleiro")
void mostra(void){
  int i;
  printf("  [");
  for(i=0;i<m;i++)
    printf(" %-d ",monte[i]);
  printf("]\n");
}

// O utilizador joga
void utente(void){
  int n,i,a;
  printf("Escolha o tamanho de um monte a dividir: ");
  scanf("%d",&n);
  for(i=0;i<m;i++)
    if(monte[i]==n && n>=3){
      printf("Qual o tamanho de uma das divisões? ");
      scanf("%d",&a);
      monte[i]=a;
      monte[m++]=n-a;
      return;
    }
}

int bloqueado(void){
  int i;
  for(i=0;i<m;i++)
    if(monte[i]>=3)
      return(0);
  return(1);
}

// E agora o pricipal!

  struct jogada{
    int imonte;  // Número do monte jogado, -1 se
                 // não existe jogada garantidamente vitoriosa
    int parte;   // Uma das partes da divisão
  } joga();      // A principal função
```

```
// O computador joga
void computador(void){
  int i,t;
  struct jogada r;
  if((r=joga()).imonte>=0){    // Está ganho!
    printf(" * ");             // Estrelinha da vitória...
    t = monte[r.imonte];
    monte[r.imonte] = r.parte;
    monte[m++] = t-r.parte;
    return;
  }
  for(i=0;i<m;i++)    // Uma jogada possível... à espera que ele jogue mal
    if(monte[i]>=3){
      monte[i]--;
      monte[m++]=1;
      return;
    }
}

/* Determina se existe uma jogada vencedora
   Se sim, coloca-a em (vm,am) e retorna 1
   Se não, retorna 0
*/
struct jogada joga(void){
  int i,a,t;
  struct jogada r;
  for(i=0;i<m;i++)
    if(monte[i]>=3){
      // Tentar todos as suas divisões do monte i
      for(a=1;a+a<monte[i];a++){
 t=monte[i];
 monte[i]=a;
 monte[m++]=t-a;
 r=joga();
 monte[i]=t;        // Repõe a situação
 m--;
 if(r.imonte<0){    // Se ele está perdido...
   r.imonte=i;
   r.parte=a;
   return(r);       // Vai ser vitória!
 }
      }
    }
  r.imonte=-1;             // Pode ser derrota
  return(r);
}
```