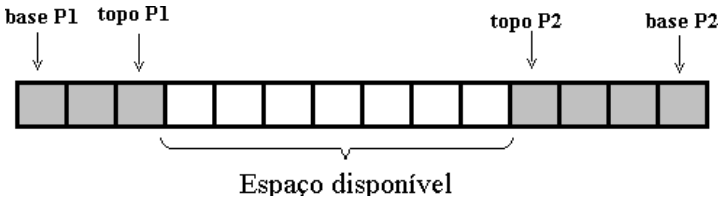
Alocação Seqüencial de Múltiplas Pilhas

Quando mais de uma pilha de elementos de mesmo tipo são utilizadas, ao invés de um array (vetor) para cada pilha, utiliza-se um array comum para todas as pilhas, fazendo com que o espaço disponível de uma seja utilizado pela outra.

Caso 1: Duas Pilhas



Características:

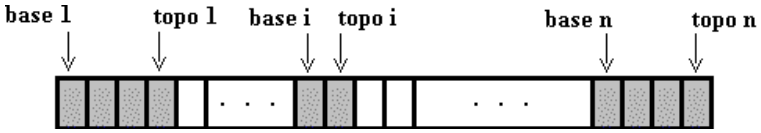
- Overflow ocorre apenas se o número total de elementos de ambas as pilhas exceder MaxTam (ou seja: topoP1 = topoP2-1)
- Base de cada pilha fica numa posição determinada na inicialização:
  - `TipoPilhaDupla* p = (TipoPilhaDupla*) malloc (sizeof (TipoPilhaDupla));`
  - `P->Pilha1.Base = 0;`
  - `P->Pilha1.Topo = -1; /* cresce para a direita */`
  - `P->Pilha2.Base = MaxTam -1;`
  - `P->Pilha2.Topo = MaxTam; /* cresce para a esquerda */`

Definição da Pilha Dupla de inteiros (sugestão):

```
#define MaxTam 1000
typedef int TipoItem;
typedef struct{
   int Topo, Base;
}IndicePilha;
typedef struct {
       TipoItem* Item[MaxTam];
       IndicePilha Pilha1, Pilha2;
} TipoPilhaDupla;
```

Caso 2: N pilhas

Quando mais de duas pilhas são alocadas no mesmo array, não é mais possível deixar as bases de cada pilha fixadas.



Nas pilhas múltiplas a inicialização é feita usando *inicialização equilibrada*, ou seja, dividindo proporcionalmente os MaxTam registros entre as N pilhas.

Sejam as pilhas i tais que:

- Base i contém o índice da posição anterior ao primeiro elemento da pilha i
- Topo i contém o índice da posição do último elemento armazenado na pilha i
- Pilha i vazia: Topo i = Base i
- Pilha i cheia: Topo i = Base (i+1)

Definição de Pilha múltipla:

```
#define MaxTam  1000
#define N 10
typedef int TipoItem;
typedef struct{
   int Topo, Base;
}IndicePilha;
typedef struct {
       TipoItem* Item[MaxTam];
       IndicePilha Pilha[N];
} TipoPilhaMultipla;
```