

# MC-102 — Aula 23

## Ponteiros III

Instituto de Computação – Unicamp

5 de Outubro de 2015

# Roteiro

1 Exemplo de Ponteiros e Alocação Dinâmica

2 Exercício

# Exemplo de Ponteiros e Alocação Dinâmica

Vamos criar uma aplicação que representa um conjunto de inteiros onde as seguintes operações podem ser realizadas:

- Inclusão de um elemento.
- Exclusão de um elemento.
- Impressão do conjunto.

# Exemplo de Ponteiros e Alocação Dinâmica

O conjunto será implementado utilizando-se um vetor.

- O tamanho do vetor deve se ajustar automaticamente: se elementos são inseridos devemos “aumentar” o vetor associado ao conjunto para incluir novos elementos; se elementos forem removidos devemos “diminuir” tal vetor.
- Temos duas variáveis associadas ao vetor:
  - ▶ **size**: denota quantos elementos o vetor possui (número de elementos do conjunto).
  - ▶ **maxSize**: denota o tamanho do vetor alocado para o conjunto.
- Vamos implementar o conjunto como um registro:

```
typedef struct Set{  
    int *data; //vetor que armazena elementos do conjunto  
    int size;  //número de elementos armazenados  
    int maxSize; //tamanho do vetor alocado data  
}Set;
```

# Exemplo de Ponteiros e Alocação Dinâmica

Temos as seguintes regras para ajuste do tamanho do vetor **data**:

- O vetor deve ter tamanho no mínimo igual a 4.
- Se o vetor ficar cheio, então devemos alocar um novo vetor com o dobro do tamanho atual.
- Se o número de elementos armazenados no vetor for menor do que  $1/4$  do tamanho do vetor, então devemos alocar um novo vetor com metade do tamanho atual.

```
typedef struct Set{  
    int *data; //vetor que armazena elementos do conjunto  
    int size;  //número de elementos armazenados  
    int maxSize; //tamanho do vetor alocado data  
}Set;
```

# Exemplo de Ponteiros e Alocação Dinâmica

Implementaremos as seguintes funções:

- **void initSet(Set \*a):** inicializa o conjunto com vetor de tamanho 4, e **size=0**.
- **void endSet(Set \*a):** libera memória alocada para o conjunto.
- **void printSet(Set \*a):** imprime elementos do conjunto.
- **int contains(Set \*a, int e):** retorna posição **i** do elemento **e** no vetor **data**, ou **-1** caso o elemento não pertença ao conjunto.

# Exemplo de Ponteiros e Alocação Dinâmica

Implementaremos as seguintes funções:

- **void addSet(Set \*a, int e)**: adiciona elemento no conjunto caso este já não pertença ao mesmo.
- **void removeSet(Set \*a, int e)**: remove elemento do conjunto caso este pertença ao mesmo.

# Exemplo de Ponteiros e Alocação Dinâmica

As funções **initSet** e **endSet** são:

```
//tamanho do vetor inicial é 4
void initSet(Set *a){
    a->data = malloc(4*sizeof(int));
    a->size = 0;
    a->maxSize = 4;
}

void endSet(Set *a){
    free(a->data);
}
```

# Exemplo de Ponteiros e Alocação Dinâmica

As funções **printSet** e **contains** são:

```
void printSet(Set *a){
    int i;
    printf("Impri. conj. de tamanho %d (tam. máx. vetor: %d)\n",a->size, a->maxSize);
    for(i=0; i<a->size; i++){
        printf("%d, ", a->data[i]);
    }
    printf("\nFim conjunto\n");
}
```

```
//retorna posição do elemento no vetor
//ou -1 se não estiver no vetor
int contains(Set *a, int e){
    int i;
    for(i=0; i<a->size; i++){
        if(a->data[i] == e)
            return i;
    }
    return -1;
}
```

# Exemplo de Ponteiros e Alocação Dinâmica

A função **addSet** é:

```
//adiciona elemento e em a
void addSet(Set *a, int e){
    if(contains(a, e) != -1) //se já estiver no conjunto sai da função
        return;

    if(a->size < a->maxSize){ //se tiver espaço para incluir o elemento
        a->data[a->size] = e; //inclui no final
        a->size++; //ajusta número de elementos armazenados
    }else{ //precisamos alocar um espaço maior
        .
        .
        .
    }
}
```

# Exemplo de Ponteiros e Alocação Dinâmica

A função **addSet** é:

```
//adiciona elemento e em a
void addSet(Set *a, int e){
    .
    .
    .
}else{ //precisamos alocar um espaço maior
    int *aux = malloc(2*(a->maxSize)*(sizeof(int)));
    int i;
    for(i=0; i<a->maxSize; i++) //salva dados em aux
        aux[i] = a->data[i];

    free(a->data); //libera memória não mais necessária

    a->data = aux; //data terá novo vetor com dobro do tamanho
    a->maxSize = 2*(a->maxSize);

    a->data[a->size] = e;
    a->size++;
}
}
```

# Exemplo de Ponteiros e Alocação Dinâmica

A função **removeSet** é:

```
//remove elemento. Caso número de dados no conjunto seja < 1/4*maxSize
//diminui tamanho do vetor pela metade
void removeSet(Set *a, int e){
    int i;
    i = contains(a,e);
    if(i == -1)//elemento não pertence
        return;

    //copia dados depois de i uma posição para trás
    for(; i< (a->size)-1 ; i++){
        a->data[i] = a->data[i+1];
    }
    a->size--;

    //se tamanho do vetor for > 4 e vetor estiver menos de 1/4 ocupado
    //devemos diminuir tamanho do vetor pela metade
    if((a->size < (0.25 * a->maxSize)) && (a->maxSize > 4) ){
        .
        .Exercício!
        .
        .
    }
```

# Exemplo de Ponteiros e Alocação Dinâmica

Com as funções implementadas podemos executar o exemplo:

```
int main(){
    Set a; int i;
    initSet(&a);

    for(i=0; i<200; i++){
        addSet(&a, i);
    }
    printSet(&a);

    removeSet(&a, 14);
    printSet(&a);

    for(i=0; i<150; i++){
        removeSet(&a, i);
    }
    printSet(&a);

    for(i=0; i<200; i++){
        removeSet(&a, i);
    }
    printSet(&a);
    endSet(&a);
}
```

# Exercício

Implemente a função de remoção de um elemento de um conjunto de tal forma que se o número de elementos no conjunto for menor do que  $\frac{1}{4}\mathbf{maxSize}$ , então o tamanho do vetor alocado para o conjunto deve ter tamanho igual a metade do anterior.