UNIFEI - Universidade Federal de Itajubá
IESTI - Instituto de Engenharia de Sistemas e Tecnologia da Informação

8º Laboratório de Estrutura de Dados (ECOP12)

Prof. Enzo Seraphim e Prof. Otávio Gomes

Alocação Dinâmica

Na alocação dinâmica podemos alocar espaços durante a execução de um programa, ou seja, a alocação dinâmica é feita em tempo de execução. Isto é bem interessante do ponto de vista do programador, pois permite que o espaço em memória seja alocado apenas quando necessário. Além disso, a alocação dinâmica permite aumentar ou até diminuir a quantidade de memória alocada.

Extraído de: Recurso Educacional Aberto para o Ens. de Alg. e Estr. de Dados (https://gabrielbueno072.github.io/rea-aed/aula_aloc.html)

Definição do Nó

```c
typedef struct no{
   int conteudo;
   struct no* proximo;
} tNoSimples;
//ponteiro para primeiro elemento
tNoSimples* primeiro = NULL;
//quantidade elementos na lista
int total = 0;
```

Criação e remoção de elementos

```c
int main(int argc, char ** argv){
  tNoSimples* novo=
    (tNoSimples*)malloc(sizeof(tNoSimples));
  strcpy(novo->animal,"pato");
  novo->proximo=NULL;
  free(novo);
  return 0;
}
```

Lista Dinâmica

Uma lista encadeada é uma estrutura de dados que representa um conjunto de dados organizados em ordem linear e dinâmica. Ela é composta por células também chamadas de nós (aqui adotaremos essa nomenclatura), que utilizando um ponteiro apontam para o próximo elemento da lista, e seu último elemento aponta para NULL, sinalizando que não existe um próximo elemento. Para que uma lista encadeada exista, basta guardar seu primeiro elemento. O primeiro é ligado no segundo, que é ligado no terceiro etc.

Extraído de: Recurso Educacional Aberto para o Ens. de Alg. e Estr. de Dados (https://gabrielbueno072.github.io/rea-aed/aula_list.html)

Inserção

```c
bool insereListaDinamica(int indice, int elemento){
  if((indice>=0)&&(indice<total)){
    tNoSimples* novo=(tNoSimples*)malloc(
                    sizeof(tNoSimples));
    novo->conteudo=elemento;
    if(indice==0){
      novo->proximo=primeiro->proximo;
      primeiro=novo;
    }else{
      tNoSimples* anterior=primeiro;
      int i=0;
      while(i<indice-1){
        anterior=anterior->proximo;
        i++;
      }
      novo->proximo=anterior->proximo;
      anterior->proximo=novo;
    }
    total++;
    return true;
  }else{ return false; }
}
```

Busca por valor

```c
//procura elemento no vetor
tNoSimples * buscaListaDinamica(int
valor){
  tNoSimples* atual=primeiro;
  while((atual!=NULL)&&(atual-
>conteudo!=valor)){
    atual=atual->proximo;
  }
  return atual;
}
```

Elemento por índice

```
//retorna o elemento que está no índice
tNoSimples* elementoListaDinamica(int
indice){
  if((indice>=0)&&(indice<total)){
    tNoSimples* atual=primeiro;
    int i=0;
    while(i<indice){
        atual=atual->proximo;
        i++;
    }
    return atual;
  }else{
    return NULL;
  }
}
```

Imprimir elementos

```
void imprimirListaDinamica(){
  tNoSimples* atual=primeiro;
  while(atual!=NULL){
    printf("[%d]",atual->conteudo)
    atual=atual->proximo;
  }
}
```

Remoção

```
bool removeListaDinamica(int indice){
  if((indice>=0)&&(indice<=total)){
    tNoSimples* atual;
    if(indice==0){
      atual=primeiro;
      primeiro=atual->proximo;
    }else{
      tNoSimples* anterior=primeiro;
      int i=0;
      while(i<indice-1){
        anterior=anterior->proximo;
        i++;
      }
      atual=anterior->proximo;
      anterior->proximo=atual->proximo;
    }
    free(atual);
    return true;
  }else{ return false; }
}
```

1) Faça um programa que cadastre 5 produtos. Para cada produto devem ser cadastrados código do produto, preço e quantidade estocada. Os dados devem ser armazenados em uma lista simplesmente encadeada e não ordenada. Posteriormente, receber do usuário a taxa de desconto (ex.: digitar 10 para taxa de desconto de 10%). Aplicar a taxa digitada ao preço de todos os produtos cadastrados e finalmente mostrar um relatório com o código e o novo preço. O final desse relatório deve apresentar também a quantidade de produtos com quantidade estocada superior a 500.

Resolução: https://github.com/student072/Exercicios-REAAED/blob/master/Listas/exercicio1.c

2) Faça um programa que cadastre 8 funcionários. Para cada funcionário cadastrado devem ser armazenados nome e salário. Os dados devem ser armazenados em uma lista simplesmente encadeada e ordenada, de forma decrescente pelo salário do funcionário. Posteriormente, programa deve mostrar:

- O nome do funcionário que tem o maior salário (em caso de empate mostrar todos).
- A média salarial de todos os funcionários juntos;
- A quantidade de funcionários com salário superior a um valor fornecido pelo usuário. Caso nenhum satisfaça essa condição, mostrar uma mensagem.

3) Faça um programa que cadastre 5 alunos. Para cada aluno devem ser cadastrados nome e nota final. Os dados devem ser armazenados em uma lista duplamente encadeada e não ordenada. Em seguida, o programa deve mostrar apenas o nome dos alunos aprovados, ou seja, com nota final de no mínimo 7. Se nenhum aluno estiver aprovado, mostrar uma mensagem.