E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Estruturas de Dados e Algoritmos Fundamentais

CÓDIGO: 21046

DOCENTE: Paulo Shirley e Paulo Quaresma

A preencher pelo estudante

NOME: Pedro Pereira Santos

N.º DE ESTUDANTE: 2000809

CURSO: Licenciatura em Engenharia Informática

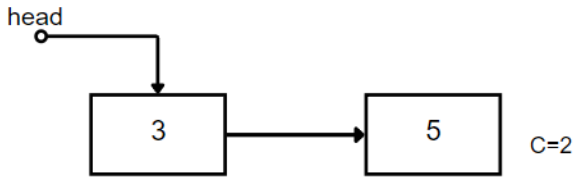
DATA DE ENTREGA: 12/04/2023

TRABALHO / RESOLUÇÃO:

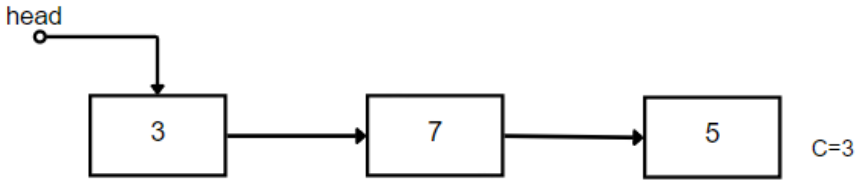
1.

1.1

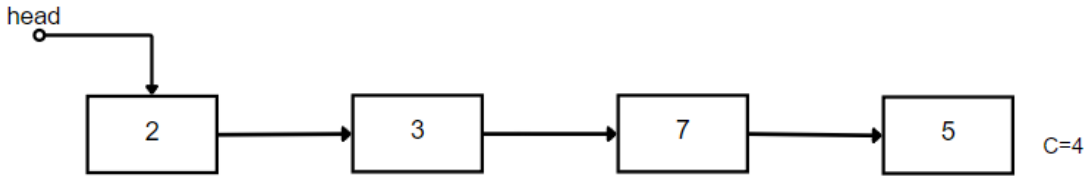
1.1.1 Inserir no fim da lista os itens 3, 5 por esta ordem



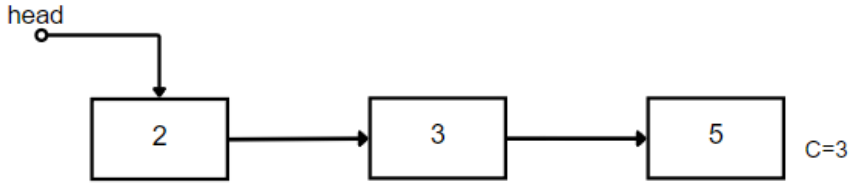
1.1.2 Inserir na posição 1 o item 7.



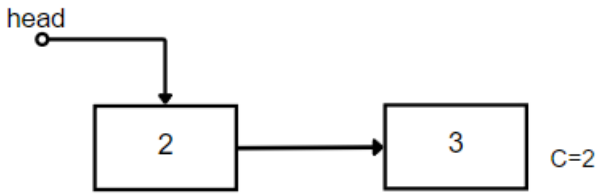
1.1.3 Inserir no início da lista o item 2.



1.1.4 Remover da lista o item 7.



1.1.5 Remover o último item da lista.



1.3

Para executar esta operação, precisamos de percorrer a lista toda à procura da posição em questão.

Temos que “head” é o primeiro nó da lista e “tail” o último.

Caso a lista esteja vazia, o algoritmo não executa, avisando o utilizador que a lista está vazia. O mesmo acontece quando a posição seja superior ao número de nós presentes na lista (pos1>n) ou caso a posição indicada seja negativa (pos1<0), indicando, neste caso, que a posição é inválida.

Começamos por verificar se a posição a procurar é a primeira (pos1==0), verificando, em caso positivo, se a lista contém ou não apenas um elemento.

Caso seja a primeira posição a remover, é logo removido o nó na primeira posição da lista, tendo o algoritmo, neste caso, complexidade O(1);

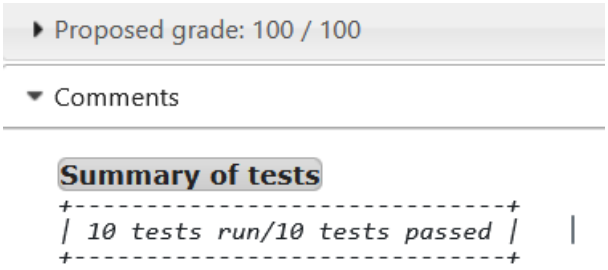
Caso, também tenha apenas um nó, “head” e “tail” ficam a apontar para um apontador nulo, caso contrário, ”head” fica a apontar para o nó logo a seguir.

Se não for a primeira posição (pos1!=0), percorremos, então, a lista à procura da posição em questão, guardando o nó precedente ao da posição a eliminar, mudando o apontador para o próximo nó, desse nó precedente, para o nó logo a seguir ao que vamos eliminar. Caso a posição seja a última da lista, mudamos tail para o nó precedente ao nó a eliminar.

Temos, caso a posição não seja a primeira, complexidade O(n), uma vez que temos de percorrer a lista toda, mas apenas uma vez.

1.2

O Código passou todos os testes, podendo vê-lo na imagem abaixo:



Temos, então, a listagem do código, pela ordem requerida:

```
/*
** file: isll.h
**
** Integer Single Linked List
** UC: 21046 - EDAF @ UAb
** e-fólio A 2022-23
**
** Aluno: 2000809 - Pedro Santos
*/


// Defina:
// em file.h as classes da estrutura de dados
// em file.cpp a implementação dos métodos das classes da estrutura de dados

// definir nó
struct INode {
  // atributos obrigatórios
  int item;           // informação em cada nó
  // outros atributos e métodos (protótipos) livres

  INode();
  ~INode();

  INode *next;
};

// definir lista simplesmente ligada
class ISll {
private:
  // atributos obrigatórios
  int n;              // dimensão atual da lista
  // outros atributos e métodos (protótipos) livres
  INode *head;
  INode *tail;
```

```
public:
  ISll();
  ~ISll();

  int getn();//devolve o valor de n
  int isEmpty(); // verifica se a lista está vazia

  void insert_0(int); //Insere no inicio o item
  void insert_end(int);//Insere no fim

  void print_0(); //imprime 1º item "Lista(0)= %d\n"
  void print_end();//imprime ultimo item "Lista(end)= %d\n"
  void print();//imprime toda a lista "Lista= %d %d ...\n"


  void delete_0();//remove 1º no
  void delete_end();//remove ultimo no
  void delete_pos(int);//remove posição x (pos 0 = 1ª)
  void clear();//remove todos os nos

  void dim();//imprime nro de items na lista "Lista tem %d itens\n"

  void find(int);//Imprime a posição da 1ª ocurrencia do item - "Item %d na posicao %d\n" ou "Item %d nao encontrado!\n"
  void find_max();//Imprime a primeira ocorrência do maior item na lista e imprime a sua posição "Max Item %d na posicao %d\n"

  void invert_range(int,int);//Inverte a ordem entre pos1 e pos2 (criar lista aux, remover e reinserir os intems em causa)

};

// EOF



/*
** file: isll.cpp
**
** Integer Single Linked List
** UC: 21046 - EDAF @ UAb
** e-fólio A 2022-23
**
** Aluno: 2000809 - Pedro Santos
*/

// Defina:
// em file.h as classes da estrutura de dados
// em file.cpp a implementação dos métodos das classes da estrutura de dados
```

```
#include <iostream>
#include "isll.h"



INode::INode() { //Constructor No
   next = nullptr;
}

INode::~INode() { //Destructor No
    next = nullptr;
    item = 0;
}

ISll::ISll() { //Constructor Lista
      head = tail = nullptr;
      n=0;
   }

ISll::~ISll() { //Destructor Lista
    for (INode *aux; n!=0; n--) {
        aux = head->next;
        delete head;
        head = aux;
    }
}

int ISll::isEmpty() {// verifica se a lista está vazia
   return n == 0;
}


int ISll::getn() {
    return n;
} //devolve o valor de n

void ISll::insert_0(int item){

    INode *aux = new INode(); //cria novo no
    aux->item=item;
    aux->next = head;
    head = aux; //na head da lista

    if(isEmpty()) //caso a lista esteja vazia, o no e tanto head como tail
       tail=head;

    n++;
} //Insere no inicio o item
```

```
void ISll::insert_end(int item){

    INode *aux = new INode(); //cria novo no
    aux->item=item;
    aux->next=nullptr;

    if(isEmpty()){//caso esteja vazia, o no é tanto head como tail
       tail=aux;
       head=tail;
    }
    else{
    tail->next=aux;
    tail=aux; // na tail da lista
    }

    n++;
}//Insere no fim

void ISll::print_0(){
    std::cout << "Lista(0)= " << head->item << std::endl;

} //imprime 1º item "Lista(0)= %d\n"

void ISll::print_end(){
    std::cout << "Lista(end)= " << tail->item << std::endl;
}//imprime ultimo item "Lista(end)= %d\n"

void ISll::print(){

    INode *aux = head;

    std::cout << "Lista=";

    for(;aux!=nullptr;aux=aux->next)//enquanto o nó nao for nulo(Enquanto nao
passar o fim da lista) imprime a lista
       std::cout << " " << aux->item;

    std::cout << std::endl;
}//imprime toda a lista "Lista= %d %d ...\n"


void ISll::delete_0(){

    INode *aux;

    if(n==1){//Se a lista so tiver 1 elemento
        head = tail = nullptr;
        delete aux;
    }
    else{
```

```
        aux = head->next;
        delete head;
        head=aux;
    }

    n--;
}//remove 1º no

void ISll::delete_end(){

    INode *aux = head;

    if(n==1){//Se a lista so tiver 1 elemento
        head = tail = nullptr;
        delete aux;
    }
    else{
        for(;aux->next!=tail;aux=aux->next);

        delete tail;
        tail = aux;
        tail->next = nullptr;
    }

    n--;
}//remove ultimo no

void ISll::delete_pos(int pos){

    INode *aux = head;


    if(pos==0){//Se a posicao for 0(a head) elimina head
        if(n!=1){
            head=head->next;
        }
        else{//Se a lista so tiver 1 elemento
            head = tail = nullptr;
        }
        delete aux;
    }
    else{//Caso contrario, procura essa posicao
        INode *tmp=aux;

        for(int i=0;i!=pos;i++){
            aux=tmp;
            tmp=tmp->next;
        }

        aux->next = tmp->next;//e elimina-a
```

```
        delete tmp;

        if(pos==n-1)//Se a posicao for a ultima da lista, atualizar o apontador da tail par ao precedente
            tail=aux;
    }
    n--;
}//remove posição x (pos 0 = 1ª)

void ISll::clear(){

    INode *aux=head;
    INode *tmp;

    while(aux!=nullptr)//Enquanto houver nos
    {
        tmp=aux;
        aux=aux->next;
        delete tmp;//elimina-os
    }

    n=0;//mete o numero de elementos igual a 0, pois foram todos eliminados
}//remove todos os nos

void ISll::dim(){
    std::cout << "Lista tem " << n << " itens" << std::endl;
}//imprime nro de items na lista "Lista tem %d itens\n"

void ISll::find(int itm){
    INode *aux=head;

    for(int i=0;aux!=nullptr;aux=aux->next,i++)//Procura toda a lista
    {
        if(aux->item==itm){//caso encontre (logo na primeira ocurrencia) imprime a informacao e retorna, para nao gastar tempo desnecessario
            std::cout << "Item " << itm << " na posicao " << i <<  std::endl;
            return;
        }
    }

    std::cout << "Item " << itm << " nao encontrado!" << std::endl;//caso contrario, retorna informacao que nao encontrou
}//Imprime a posição da 1ª ocurrencia do item - "Item %d na posicao %d\n" ou "Item %d nao encontrado!\n"

void ISll::find_max(){
    INode *aux = head;
    int pos=0, itm = head->item; // inicia itm com o primeiro itm, para guardar valor para comparar
```

```
    for(int i=0;aux!=nullptr;aux=aux->next,i++)//Procura toda a lista
    {
        if(aux->item>itm){//Caso item a testar seja maior
            itm=aux->item;//Guarda item sempre que encontrar um item maior que o guardado
            pos=i;//e a sua posicao
        }
    }

    std::cout << "Max Item " << itm << " na posicao " << pos << std::endl;

}//Imprime a primeira ocorrência do maior item na lista e imprime a sua posição "Max Item %d na posicao %d\n"

void ISll::invert_range(int pos1,int pos2){

    ISll L_aux;//cria lista auxiliar
    INode *aux=head;//copia no inicial
    INode *checkpoint=head; /*inicializar checkpoint, no caso de pos1=0. Serve para guardar um ponto para continuar depois de passar para a lista aux.*/
    int i;

    for(i=0;i<pos1;i++){//procura posicao 1
        checkpoint=aux;
        aux=aux->next;
    }

    for(int j=i;j<=pos2;j++){//guarda no a inverter, ja na posicao requerida
        L_aux.insert_0(aux->item);
        aux=aux->next;
    }

    INode *add;

    if(pos1!=0)/*caso a posicao nao seja 0, checkpoint e o no imediatamente antes do primeiro no dentro do "range" para inverter */
        add=checkpoint->next;/*Logo, queremos o no imediatamente a seguir para comecar a trocar*/
    else/*caso a posicao seja 0, queremos logo comecar a mudar na posicao de head*/
        add=head;

    for(;i<=pos2;i++){//com a lista auxiliar, copia-se os valores para a lista principal
        add->item=L_aux.head->item;
        add=add->next;
        L_aux.delete_0();//apagando no a no a lista auxiliar
    }

}//Inverte a ordem entre pos1 e pos2 (criar lista aux, remover e reinserir os items em causa)
```

```
// EOF


/*
** file: main-isll.cpp
**
** Integer Single Linked List
** UC: 21046 - EDAF @ UAb
** e-fólio A 2022-23
**
** Aluno: 2000809 - Pedro Santos
*/

// Defina neste ficheiro:
//   A entrada/saída de dados
//   As instâncias da classe da estrutura de dados
//   A implementação dos comandos através dos métodos da classe
//   Código auxiliar
//   Não utilize variáveis globais

#include <iostream>
#include <sstream>
#include "isll.h"
using namespace std;


int main()
{
  ISll list;   // exemplo
  std::string input,command; //string de recepcao de input e recepcao do comando
  int arg1,arg2; // argumentos a usar nas funcoes
  stringstream ss;

//Ler 1 linha, Processar 1 linha (while !EOF) Sugerencia de ler para uma string e analisar se o primeiro caracter é # ou nao. Se não for
//Criar uma stringstream e usar >> para obter o nome do comando. Se for linha em branco, falha. (Salta espaços, tabs e new lines antes de efetuar uma leitura)


   while(getline(cin,input)){ /*Utiliza-se a condição de se a leitura foi bem sucessida para fazer um loop para ler todo o input*/

   if(input[0]!='#'&&!input.empty())/*Verifica-se se o primeiro caracter é # ou se é uma linha em branco, continuado o programa em caso negativo*/
   {
      ss << input; /*passa-se o input para a string stream*/
      ss >> command; /* passa-se apenas o comando com o uso de >> para a var. que guarda o comando*/
```

```
/*Para todos os comandos, compara-se a string guardada na var command com a palavra que interpreta o comando, executando-o caso sejam iguais*/
	if(!command.compare("insert_0")){
	   while(ss>>arg1) /* caso haja mais de 1 argumento, faz um ciclo para executar varias vezes o comando*/
	   list.insert_0(arg1);
	}
	else if(!command.compare("insert_end")){
	   while(ss>>arg1) /* caso haja mais de 1 argumento, faz um ciclo para executar varias vezes o comando*/
	   list.insert_end(arg1);
	}
	else if(!command.compare("dim"))
	   list.dim();
	else if(!list.isEmpty()){/* como os comandos a baixo desta condicao sao os unicos que precisas de que haja elementos, mete-se a condição neste nivel*/
	   if(!command.compare("print_0"))
	      list.print_0();
	   else if(!command.compare("print_end"))
	      list.print_end();
	   else if(!command.compare("print"))
	      list.print();
	   else if(!command.compare("delete_0"))
	      list.delete_0();
	   else if(!command.compare("delete_end"))
	      list.delete_end();
	   else if(!command.compare("clear"))
	      list.clear();
	   else if(!command.compare("find")){
	      ss >> arg1;//vai buscar o argumento a usar
	      list.find(arg1);
	      }
	   else if(!command.compare("find_max"))
	      list.find_max();
	   else if(!command.compare("delete_pos")){
	      ss >> arg1;//vai buscar o argumento a usar
	      if(arg1>list.getn()||arg1<0)//Verifica-se se a posicao é valida
	         cout << "Comando " << command << ": Posicao invalida!\n";
	      else
	         list.delete_pos(arg1);
	      }
	   else if(!command.compare("invert_range")){
	      ss >> arg1 >> arg2;//vai buscar os argumentos a usar
	      if(arg1>list.getn()||arg2>list.getn()||arg1>arg2||arg1<0)//Verifica-se  se a posicao é valida
	         cout << "Comando " << command << ": Posicao invalida!\n";
	      else
	      list.invert_range(arg1,arg2);
```

```
            }
        }
        else//caso a lista esteja vazia, reporta-o
            cout << "Comando " << command << ": Lista vazia!\n";
    ss.clear();//Limpa a
    ss.str("");//stringstream
  }
}

  return 0;
}

// EOF
```