**Nomes:**

**Henrique Sartori Siqueira 19240472**

**Rafael Silva Barbon 19243633**

**EX1:** Qual a diferença entre passagem de parâmetros por valor e referência?

**Resposta:** Quando um parâmetro é passado por valor, estamos copiando o valor da variável do programa principal para um variável local da função, ou seja, não é possível alterar o valor da variável do programa principal na função, já que apenas o conteúdo da variável é passado para função.

Na passagem por parâmetro, a função recebe o endereço da variável declarada na main e não o seu conteúdo, com isso a função consegue, além de acessar o conteúdo da variável passada como parâmetro, alterar o seu valor, já que a função possui o endereço de memória da variável declarada no programa principal.

**EX2:** Explique por que não é possível igualar uma lista à outra apenas igualando dois ponteiros.

**Resposta:** Ao utilizar um ponteiro para indicar uma lista significa que tal ponteiro guarda o endereço do início de tal lista e não as informações armazenadas na lista em questão. Caso não haja esse ponteiro apontando para o início da lista, todos os dados alocados na memória seriam perdidos na execução do programa, porém eles ainda continuariam alocados e ocupando espaço da memória RAM. Assim, ao igualar dois ponteiros, há apenas a criação de um novo ponteiro que faz a função de guardar o endereço da primeira partição da lista, sendo este útil para loops. Para que uma lista seja igual a outra, é necessário que haja a alocação de uma outra lista e copiar as informações de cada subconjunto da lista para a nova lista, assim haverá uma igualdade de listas, ou seja, as informações seriam iguais, mas os ponteiros indicando o início da lista teriam valores diferentes, pois cada lista ocuparia um lugar distinto na memória RAM.

**EX3:** O que o programa abaixo exibe?

int main (void) {

typedef struct{

int dia, mes, ano;

} data;

printf ("sizeof (data) = %d\n", sizeof(data)); }

**Resposta:** O programa imprime o tamanho do tipo de variável data, como a sua estrutura possui três variáveis do tipo inteiro, cujo o tamanho de cada inteiro é de 4 bytes, o programa imprime o valor 12, ou seja, o tamanho do tipo data é de 12 bytes.

**EX4:** O código abaixo está correto? Justifique.

void troca (int\*i, int\*j) {

int \*temp = \*i;

\*i = \*j;

\*j = \*temp;

}

**Resposta:** O código acima não está correto, pois a utilização da variável "temp" é incorreta, a função considera "temp" como um ponteiro que recebe o valor contido em "i" e nao seu endereço, assim, ao executar o programa haverá o erro de falha de segmentação, para corrigir tal erro, deve-se inicializar e realizar a manipulação da mesma sem o ponteiro (\*), pois o objetivo da função e de trocar os valores passados por parâmetro e, por isso, é necessário que a variável declarada seja do tipo inteiro para que consiga receber o valor apontado do ponteiro inteiro "i".

**EX5:** Até o momento, todos os nossos exercícios foram utilizando uma estrutura de inteiros. Assim, crie uma biblioteca chamada LISTA\_STRING.h que deverá permitir a manipulação de uma estrutura que possua um vetor de char (string) de 50 caracteres. Esta biblioteca deverá ter todas as funções que visualizamos para a lista de inteiros.

**Resposta:**

#ifndef LISTA\_STRING\_h\_INCLUDED

#define LISTA\_STRING\_h\_INCLUDED

/\*

Insere\_LS - Insere no inicio da lista

Vazia\_LS - Retorna 1 caso a lista esteja vazia, caso contrario, 0

Imprime\_LS - Imprime todas as strinfs contidas na lista

Retira\_LS - Retira parte da lista a partir da posiçao fornecida pelo usuario

Desaloca\_LS - Libera a memoria dos elementos previamente alocados na lista

\*/

typedef struct listast{

char string[50];

struct listast \*prox;

}Lista\_String;

Lista\_String\* Cria\_LS(){

return NULL;

}

void Insere\_LS(Lista\_String \*\*recebida, char string[]){

Lista\_String \*novo = (Lista\_String \*)malloc(sizeof(Lista\_String));

strcpy(novo->string,string);

novo->prox = \*recebida;

\*recebida = novo;

}

int Vazia\_LS(Lista\_String \*recebida){

if(recebida == NULL)

return 1;

return 0;

}

void Imprime\_LS(Lista\_String \*recebida){

printf("\n\tLista:");

for(Lista\_String \*aux = recebida; aux != NULL; aux = aux->prox){

printf("\n\t%s",aux->string);

}

}

Lista\_String \* Busca\_LS(Lista\_String \*recebida,char alvo[]){

Lista\_String \*aux = recebida;

int conf;

while(aux != NULL){

conf = strcmp(aux->string,alvo);

if(conf == 0)

return aux;

aux = aux->prox;

}

return NULL;

}

void Retira\_LS(Lista\_String \*\*recebida, int pos){

Lista\_String \*aux = (\*recebida), \*aux2 = Cria\_LS();

for(int count = 1; aux != NULL; aux2 = aux, aux = aux->prox, count++){

if(count == pos){

if(count == 1)

(\*recebida) = (\*recebida)->prox;

else

aux2->prox = aux->prox;

free(aux);

break;

}

}

}

void Desaloca\_LS(Lista\_String \*Recebida){

Lista\_String \*aux = Recebida, \*aux2;

while(aux != NULL){

aux2 = aux->prox;

free(aux);

aux = aux2;

}

}

#endif

**EX6:** Utilizando a biblioteca criada no exercício anterior, faça um programa que solicite ao usuário uma sequencia alternada de letras e números. Seu programa deverálistar as letras na ordem em que foram inseridas e os números na ordem inversa. Exemplos: Entrada: A1E5T7W8G –Saída: AETWG8751 Entrada: 3C9H4Q6 –Saída: CHQ6493

**Resposta:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "LISTA\_STRING.h"

#define tam 50

int eh\_numero(char c)//Verifica se o caracter é um número

{

if(c >= 48 && c <= 57)

return 1;

return 0;

}

void lista\_ordem(Lista\_String \*L)

{

Lista\_String \*aux;

char C[tam], novo[tam];

int len, i, j=0;

for(aux=L; !Vazia\_LS(aux); aux = aux->prox)

{

j=0;

strcpy(C,aux->string);

len=strlen(C);

for(i=0; i<len; i++)

if(!eh\_numero(C[i]))

novo[j++]=C[i];

for(i=len-1; i>=0; i--)

if(eh\_numero(C[i]))

novo[j++]=C[i];

novo[j]='\0';

strcpy(aux->string, novo);

}

}

int main()

{

Lista\_String \*L = Cria\_LS();

char info[tam];

int conf;

do{

printf("\nInsira Letras e numeros de maneira alternada:(digite BREAK para sair):");

fflush(stdin);

scanf("%s", info);

strupr(info);

conf=strcmp(info,"BREAK");

if(conf != 0)

Insere\_LS(&L, info);

}while(conf != 0);

system("clear");

printf("\nLista Lida:");

Imprime\_LS(L);

lista\_ordem(L);

printf("\n\nLista modificada:");

Imprime\_LS(L);

Desaloca\_LS(L);

return 0;

}