UDESC – Universidade do Estado de Santa Catarina
TADS – Tecnologia em Análise e Desenvolvimento de Sistemas
EDA0001 – Estruturas de Dados
Professor Rui Tramontin

Lista de Exercícios 1 – Ponteiros

(a) <u>Passagem de Parâmetros por Referência</u>: para as questões de 1 a 5, mesmo quando não mencionado, é preciso implementar um programa que utiliza a função solicitada.

1) Escreva uma função que troca os valores entre duas variáveis do tipo *float*. Faça um programa que leia duas variáveis e mostre seus valores na tela. Em seguida, troque os valores (usando a função) e mostre novamente os valores.

```
void troca_valor(float *x, float *y);
```

2) Faça uma função que calcula o perímetro e a área de um círculo, dado o raio.

```
void calcula_circulo(float raio, float *pPerimetro, float *pArea);
```

3) Faça uma função que receba um parâmetro (por valor) com o total de minutos passados ao longo do dia e receba também dois parâmetros (referência) no qual deve preencher com o valor da hora e do minuto corrente. Faça um programa que leia do teclado quantos minutos se passaram desde meia-noite e imprima a hora corrente (use a sua função).

```
void cacula_hora(int totalMinutos, int *ph, int *pm);
```

4) Escreva uma função que recebe um vetor e sua capacidade como parâmetros e precisa "retornar" o maior e o menor valores do vetor.

```
void max_min(int vet[], int tam, int *pMin, int *pMax);
```

5) Escreva um programa que determine o maior valor de um vetor bem como sua posição no vetor (índice). Tal processamento deve ser feito em uma função que recebe o vetor (do tipo *float*) e sua capacidade, e "retorna" o maior elemento e sua posição.

```
void max_vetor(float vet[], int tam, float *pMax, int *pIndice);
```

(b) Alocação Dinâmica de Memória:

6) Escreva uma função que recebe um vetor *float* ***v*** e sua capacidade ***n***, e retorne o endereço de um vetor alocado dinamicamente, cujo conteúdo seja o mesmo de ***v***, ou seja, a função retorna um *clone* do vetor ***v***. Faça o programa principal com a entrada de dados (ou um vetor fixo), chame a função e mostre o vetor resultante na tela. Protótipo da função:

```
float *clone( float *v, int n );
```

7) Escreva uma função que recebe como parâmetros uma *string* ***s*** e um inteiro ***n***, e retorna nova *string* nova contendo ***s*** repetida ***n*** vezes. Por exemplo, ***s*** = "Abc" e ***n*** = 4 tem como resultado a *string* "AbcAbcAbcAbc". Faça o programa principal chamando a função. Protótipo da função:

```
char *repetidor( char *s, int n );
```

8) Escreva um programa que aloca dinamicamente um vetor do tipo *float* e realiza a entrada de dados. Em seguida, o programa deve calcular a *média* dos valores do vetor e alocar dinamicamente um novo vetor contendo somente os valores maiores ou iguais à media. O processo pode ser feito usando *malloc()*, ou seja, fazendo a contagem, alocação e cópia dos valores. Outra alternativa consiste em usar *realloc()* para ir aumentando o espaço alocado à medida que os valores vão sendo encontrados.

9) Faça um programa que leia uma certa quantidade de inteiros que são armazenados num vetor ***v***. A quantidade deve ser definida pelo usuário, e o programa aloca espaço para ***v***. O programa deve armazenar os valores positivos em um vetor ***vp*** e o valores negativos no vetor ***vn***. Como as quantidades de valores positivos e negativos são desconhecidas, o espaço para ***vp*** e ***vn*** deve ser alocado dinamicamente. Os vetores ***vp*** e ***vn*** não devem conter zeros. Ao final, imprima os três vetores. Pode ser feito com *malloc()* ou com *realloc()*.

10) Escreva uma função que realiza a ***união*** entre dois conjuntos de inteiros contidos nos vetores ***v1*** e ***v2***. A função recebe os vetores e suas respectivas capacidades (***n1*** e ***n2***) como parâmetros de entrada e retorna o endereço do vetor alocado (contendo a união entre ***v1*** e ***v2***). Além disso, há um parâmetro passado por referência (ponteiro ***p3***), que serve para "retornar" a capacidade do vetor gerado. Faça o programa principal invocando a função. Protótipo da função:

```
int *uniao( int *v1, int n1, int *v2, int n2, int *p3 );
```