Ficha | Ponteiros e passagem de argumentos

> Utilize o *template* `CProgram_Template` disponível no *GitHub*:
>
> https://github.com/estsetubal-atad/CProgram_Template

Passagem de argumentos por referência

1. Crie uma função `void double(int *val)`que recebe um inteiro por referência e duplique o seu valor. Crie uma função main onde utilize este função.
2. Crie uma função `void swap(int *a, int *b)`que recebe dois inteiros por referência troca o conteúdo das variáveis. Crie uma função main onde utilize este função.

- Sugestão: esta função pode ser utilizada nos algoritmos de ordenação lecionados - para efetuar as trocas.

3. Crie uma função `bool mean(int arr[], int arrLength, double *mean)` que recebe um array e o seu tamanho e devolve:

- *por referência*: a média dos elementos do array;
- por retorno: `true` se a média pode ser calculada; `false`, caso contrário, e.g., se o array for "vazio";

  O valor só é devolvido por referência, se puder ser calculado; caso contrário mantém-se inalterado.

  Crie um programa que utilize esta função.

4. Implemente a função `bool arrayStats(int arr[], int arrLength, int *min, int *max, double *mean)` que devolve:
   - *por referência* os valores mínimo, máximo e média (tente reutilizar a função do exercício anterior);
   - *por retorno* `true` se o array contém elementos e as operações de seleção decorreram com sucesso; `false` se o array está vazio.
5. Forneça a documentação *doxygen* da função anterior, seguindo as convenções documentadas nos materiais da UC.

Ponteiros e tipos compostos

> Utilize o *template* `Students_Template` disponível no *GitHub*:
>
> https://github.com/estsetubal-atad/Students_Template

1. No módulo `students` adicione a função `void studentPrint(PtStudent s)` que recebe por referência um aluno e imprime a sua informação. Modifique o programa por forma a utilizar esta função para imprimir a informação de todos os alunos importados.

2. No módulo `students` adicione a função `bool studentSame(PtStudent s1, PtStudent s2)` que recebe dois alunos por referência e verifica se são "iguais", no sentido de serem a mesma "instância" (residirem no mesmo espaço de memória!).
3. No módulo `students` adicione a função `bool studentEquals(PtStudent s1, PtStudent s2)` que recebe dois alunos por referência e verifica se são "iguais" em termos dos valores dos seus membros; os membros de ambas as estruturas devem possuir os mesmos valores para serem considerados iguais (utilize `==` e `strcmp` apropriadamente).
4. Teste as duas funções anteriores para ambos os resultados possíveis.

Manipulação de ponteiros

1. Considere o seguinte código:

```c
int main(){
    char str[] = "Ponteiros";
    char *p = str;

    /* PRINT ADDR ARRAY */

    /* PRINT ADDR 'n' */

    char charT_indexed = /* A */;
    char charT_arithmetic = /* B */;

    /* LOOP INDEXED */

    /* LOOP ARITHMETIC */

    return EXIT_SUCCESS;
}
```

Utilizando apenas `p`, qual o código que preenche os comentários para:

- `PRINT ADDR` - Imprimir o endereço do array de caracteres;
- `PRINT ADDR 'n'` - Imprimir o endereço do caractere `n` contido no array (uma forma qualquer forma que funcione);
- `A` - obter o caractere `t` através de indexação;
- `B` - obter o caractere `t` através de aritmética de ponteiros;
- `LOOP INDEXED` - percorrer e imprimir todos os caracteres do array através de indexação;
- `LOOP ARITHMETIC` - percorrer e imprimir todos os caracteres do array através de aritmética de ponteiros.

2. Qual o *output* deste programa?

```c
void func(int *p, int *v){
    p++;
```

```
    *v = p[0];
    v++;
    *(v+1) = p[1];
    v[0] = *(p+2);
}

int main(){
  int v[] = {23, 6, 24, 2};
  int k[4] = {0};
  int *ptr = k;

  func(v, ptr);
  for(ptr = k; ptr < k+4; ptr++)
      printf("%d\n", *ptr);

  return EXIT_SUCCESS;
}
```