# ATIVIDADES PRÁTICAS SUPERVISIONADAS

## CST em Análise e Desenvolvimento de Sistemas

3ª Série

Programação Estruturada II

A atividade prática supervisionada (ATPS) é um método de ensino-aprendizagem desenvolvido por meio de um conjunto de atividades programadas e supervisionadas e que tem por objetivos:

- ✓ Favorecer a aprendizagem.
- ✓ Estimular a corresponsabilidade do aluno pelo aprendizado eficiente e eficaz.
- ✓ Promover o estudo, a convivência e o trabalho em grupo.
- ✓ Desenvolver os estudos independentes, sistemáticos e o autoaprendizado.
- ✓ Oferecer diferenciados ambientes de aprendizagem.
- ✓ Auxiliar no desenvolvimento das competências requeridas pelas Diretrizes Curriculares Nacionais dos Cursos de Graduação.
- ✓ Promover a aplicação da teoria e conceitos para a solução de problemas relativos à profissão.
- ✓ Direcionar o estudante para a emancipação intelectual.

Para atingir estes objetivos as atividades foram organizadas na forma de um desafio, que será solucionado por etapas ao longo do semestre letivo.

Participar ativamente deste desafio é essencial para o desenvolvimento das competências e habilidades requeridas na sua atuação no mercado de trabalho.

Aproveite esta oportunidade de estudar e aprender com desafios da vida profissional.

**AUTORIA:**

Jaqueline Brigladori Pugliesi
*Faculdade Anhanguera de Valinhos*

Jeanne Dobgenski
*Anhanguea Educacional Ltda*

Marcelo Augusto Cicogna
*Faculdade Anhanguera de Valinhos*

## COMPETÊNCIAS E HABILIDADES

Ao concluir as etapas propostas neste desafio você terá desenvolvido as competências e habilidades descritas a seguir.

- ✓ Capacidade de, com base nos conceitos adquiridos, iniciar, projetar, desenvolver, validar e gerenciar qualquer projeto de *software*.
- ✓ Competência para identificar, analisar, documentar e solucionar problemas e necessidades passíveis de solução via computação.
- ✓ Capacidade para desenvolvimento de pesquisa científica e tecnológica.
- ✓ Saber conciliar teoria e prática.

### Produção Acadêmica

- Discussões em grupo.
- Planejamento das atividades.
- Relatório virtual sobre o desenvolvimento de um pequeno aplicativo de linha de comando (Console).

### Participação

Para a elaboração desta atividade, os alunos deverão previamente organizar-se em equipes de até seis participantes, seguindo as diretrizes do tutor presencial.

### Padronização

O material escrito solicitado nesta atividade deve ser produzido de acordo com as normas da ABNT[1], com o seguinte padrão:

- Em papel branco, formato A4.
- Com margens esquerda e superior de 3cm, direita e inferior de 2cm.
- Fonte *Times New Roman* tamanho 12, cor preta.
- Espaçamento de 1,5 entre linhas.
- Se houver citações com mais de três linhas, devem ser em fonte tamanho 10, com um recuo de 4cm da margem esquerda e espaçamento simples entre linhas.
- Com capa, contendo:
  - Nome de sua Unidade de Ensino, Curso e Disciplina.
  - Nome e RA de cada participante.
  - Título da atividade.
  - Nome do professor da disciplina.
  - Cidade e data da entrega, apresentação ou publicação.

## DESAFIO

Muitos programas de computador do tipo "Console" são utilizados todos os dias. Para a plataforma *Linux*, esta é uma afirmação quase que incorporada à estrutura do próprio Sistema Operacional (SO). No SO *Windows*, os aplicativos de console, que são simples em interface, mas que desempenham tarefas muitas vezes sofisticadas. Pode-se citar desde programas de linha de comando como "*ipconfig*" que auxilia na visualização das

[1] Consultar o Manual para Elaboração de Trabalhos Acadêmicos. Unianhanguera. Disponível em: <http://www.unianhanguera.edu.br/anhanguera/bibliotecas/normas_bibliograficas/index.html>.

configurações de rede de um PC, até programas como o "*taskkill*" que auxiliam o usuário a eliminar processos ativos na memória do computador.

O objetivo desta atividade é adquirir experiência com o ciclo completo de desenvolvimento de um pequeno aplicativo de linha de comando (Console).

Sendo assim, esta atividade concentra esforços no desenvolvimento de um programa que permita contar linhas de código fonte desenvolvida em Linguagem C ou C++ e a estrutura sugerida permitirá a sua adequação para outras linguagens.

## Objetivo do Desafio

Construir um relatório virtual de um ciclo completo de desenvolvimento de um pequeno aplicativo de linha de comando (Console).

## ETAPA 1

Esta atividade é importante para você compreender os requisitos do problema e determinar uma organização inicial do código fonte que permita o desenvolvimento de uma solução.

Para realizá-la é importante seguir os passos descritos.

## PASSOS

### Passo 1 (Aluno)

Ler atentamente o desafio e os conceitos de contagem de linhas. Identificar os dois tipos principais de comentários possíveis em linguagem C: comentário de uma linha apenas, que começa com " / / "; e comentários multi-linhas determinado por " /* " e " */ ".

### Passo 2 (Equipe)

Analisar o problema e fazer sugestões de como organizar o código em uma função principal (main) e outra função que fará a contagem das linhas dado como parâmetro o nome do arquivo a ser pesquisado. Pode-se chamar esta função de ccRun.

**Observação:** ao utilizar o prefixo "cc" (de *Code Count*), ou qualquer outro que o aluno achar conveniente, facilita-se a integração do código desenvolvido nesta tarefa com outras que o programador venha a achar necessário no futuro.

Como guia para alunos e professores, os autores deste desafio determinaram dois módulos para a solução do problema.

O módulo principal foi denominado *MainCodeCount* e um módulo auxiliar com funções de auxílio ao principal chamado *LibCodeCount*. Ao mesmo tempo, foram projetadas duas funções, conforme a orientação anterior. Os cabeçalhos e os nomes dos arquivos os quais conterão o código fonte são apresentados na Figura 2.

Arquivo **MainCodeCount.c**

```
int main(int argc, char *argv[])
{...
}
```

Arquivo **LibCodeCount.h**

```
void ccRun(char* file, int* nLines, int* nLinesComment, int* nLinesEmpty,
  int noComment, int silent);
```

Arquivo **LibCodeCount.c**

```
void
ccRun(char* file, int* nLines, int* nLinesComment, int* nLinesEmpty,
  int noComment, int silent);
{...
}
```

Figura 2. Exemplo de uso e funcionamento do contador de linhas de código fonte.

**Passo 3 (Equipe)**

Estudar a modularização proposta e fazer com que seu código siga a organização sugerida nesta atividade, ou seja, que possua pelo menos dois módulos: um principal e outro auxiliar. Não será aceito na avaliação desta etapa a apresentação de apenas um arquivo de código fonte (arquivo.c).

**Passo 4 (Equipe)**

Compor, por fim, o código fonte formado por dois módulos que contenham a função principal e uma função auxiliar para o cálculo do número de linhas, considerando as opções fornecidas nesta etapa.

## ETAPA 2

Esta atividade é importante para você entender a passagem de parâmetros para um programa escrito em linguagem C.

Para realizá-la é importante seguir os passos descritos.

## PASSOS

**Passo 1 (Aluno)**

Pesquisar a utilização dos argumentos "*argc*" e "*argv*" passados como parâmetros da função principal "*main*()" de um programa escrito em linguagem C.

**Passo 2 (Equipe)**

Elaborar uma função principal que aceite os seguintes parâmetros:

**Parâmetro obrigatório.**
"Nome do arquivo". O primeiro parâmetro será o nome de um arquivo a ser pesquisado o número de linhas.

**Parâmetros opcionais.**
-h (-*help*) opção para apresentar uma explicação do uso do programa *CodeCount*.
-c (-*comment*) opção para contar o número de linhas de comentários.
-s (-*silent*) opção para não apresentar resultados na tela de comando (*prompt*).

Para auxílio dos alunos, apresenta-se um exemplo na Listagem 3 de como criar o parâmetro "–s". Neste exemplo será possível notar o uso dos argumentos "*argc*" e "*argv*". Vale lembrar que a posição zero do vetor "*argv*" possui sempre o nome do programa, neste caso "*CodeCount*".

Como se trata de criar uma forma de implementar os parâmetros opcionais, o *loop* pelo arquivo "*argv*" começa na posição 2, uma vez que a posição 1 foi reservada para conter o nome do arquivo a ser pesquisado. Assim, a chamada do programa (o símbolo ">" representa a linha de comando, ou *prompt* do sistema operacional):

```
> CodeCount Exemplo.c –c
```

Teria os seguintes argumentos passados para a função principal:

```
argc = 3
argv[0] = "CodeCount"
argv[1] = "Exemplo.c"
argv[2] = "-c"
```

Listagem 3. Exemplo de função principal com gerenciamento de parâmetros.

```
int
main(int argc, char *argv[])
{
  int i, silent;
  int nLines, nLinesComment, nLinesEmpty, count;
  silent = 0;
  nLines = 0;

  for (i=2; i<argc; i++)
  {
    silent = silent || (strstr(argv[i], "-silent"  ) != NULL);
    silent = silent || (strstr(argv[i], "-s"       ) != NULL);
  }
  printf("Code Count - Programmer's Tool\n");
  printf("  Version 1.1 - 2009\n\n");

  if (!silent)
  {
    printf("  counting for \"%s\"...\n", argv[1]);
  }

  ccRun(argv[1], &nLines, &nLinesComment, &nLinesEmpty, noCom, silent);
```

**Nota:** Observe que o exemplo testa se o usuário passou a versão curta do parâmetro "-s" ou a versão normal "*-silent*".

### Passo 3 (Equipe)

Compor, por fim, o código fonte, em complemento ao que foi entregue na etapa anterior, que apresente uma função principal capaz de processar os parâmetros e opções estabelecidos nesta etapa.

## ETAPA 3

Esta atividade é importante para você determinar a versão final do programa capaz de contar linhas e comentários, bem como utilizar a passagem de parâmetros projetada na etapa anterior.

Para realizá-la é importante seguir os passos descritos.

## PASSOS

**Passo 1 (Aluno)**

Ler o texto e fazer as atividades a seguir.

Nesta etapa, será necessário entender o conteúdo de um arquivo texto e elaborar uma estratégia para a contagem de linhas, principalmente focando o problema de contagem de linhas com comentários.

Por exemplo, a leitura de um arquivo texto pode ser interpretada como o percurso por um vetor de caracteres. Como ilustração desta ideia, veja o arquivo abaixo:

```
Arquivo
Segunda linha
fim
```

A leitura deste arquivo no disco pode ter uma representação da seguinte forma:

| A | r | q | u | i | v | o | \n | S | e | g | u | n | d | a | | l | i | n | h | a | \n | f | i | m | EOF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Notar que as linhas têm o caractere especial '\n' para delimitar o seu fim, bem como o final do arquivo é encontrado quando se processa a informação EOF (*End Of File*).

A ideia para a contagem de linhas, conforme visto na Etapa 1, baseia-se na contagem dos caracteres '\n' de um arquivo texto.

Ao mesmo tempo, a contagem de linhas com comentários pode fazer uso da mesma estratégia.

Por exemplo, o uso de variáveis auxiliares, pode-se determinar a ocorrência dos caracteres "\\", indicando que esta linha é um comentário e que, portanto, ao encontrar um caractere '\n' esta linha deva ser contada como comentário.

O mesmo vale para comentários multi-linhas. Ao se identificar os caracteres "/*" todos os caracteres '\n' encontrados antes da ocorrência de "*/" deverão ser contabilizados como linhas de comentário.

**Passo 2 (Equipe)**

Desenvolver um algoritmo que baseado na leitura caractere a caractere de um arquivo texto, contabilize o número de linhas total, o número de linhas de comentário simples e o número de linhas de comentário multi-linhas. Por exemplo, considerar o arquivo descrito a seguir:

```
// Este é um comentário de uma linha só.
for (i = 0; i < 10; i++)
{
  /* Este é um comentário de duas linhas.
     O término é ocorre na próxima linha.
  */
  printf("Valor de i %d", i);  //Um comentário aqui não deve ser computado.
}
```

**Nota:** Este arquivo possui 8 linhas. Uma linha é do tipo comentário simples, iniciado com "//". Duas linhas possuem comentário multi-linhas. Neste caso, não foi contabilizada a linha com o "*/", mas os alunos podem encontrar outra regra. É importante notar que o comentário apresentado após o comando "*printf*" não deve ser contabilizado, uma vez que a mesma linha possui código.

**Passo 3 (Equipe)**

Compor, por fim, o código fonte, em complemento ao que foi entregue na etapa anterior, que apresente uma função principal capaz de contabilizar o número de linhas total de um arquivo de código fonte em linguagem C. Em complemento, o programa deve também contabilizar o número de linhas com comentários simples, multi-linhas e o número de linhas "em branco" (sem comentários e/ou instruções de código).

## ETAPA 4

Esta atividade é importante para você validar o programa desenvolvido.
Para realizá-la é importante seguir os passos descritos.

## PASSOS

**Passo 1 (Equipe)**

Criar três tipos de arquivos de código fonte escritos em Linguagem C. Por exemplo, criar um arquivo com poucos comentários, fáceis de serem identificados. Os outros dois arquivos podem possuir um número maior de combinações e linhas de comentários. Veja, por exemplo, o código apresentado na Listagem 2.

**Passo 2 (Equipe)**

Demonstrar o uso do programa desenvolvido, bem como os resultados obtidos. Ter como objetivo apresentar ao usuário um documento simples que demonstre que seu programa funciona bem e que determina os resultados esperados.

**Passo 3 (Equipe)**

Postar no ambiente virtual, o arquivo em *Word* (extensão.doc) intitulado "Desenvolvimento de um pequeno aplicativo de linha de comando – Console" construído mediante cumprimento das quatro etapas, para a avaliação do tutor à distância.

## Livro Texto da Disciplina

DEITEL, P. J.; DEITEL, H. M. **C. :** *Como Programar*. 6ª ed. São Paulo: Pearson, 2011.

## Critérios de Avaliação

A avaliação será realizada pelo tutor à distância, que levará em consideração:

- Capacidade de síntese e organização do material desenvolvido nas etapas.
- Cumprimento e qualidade do conteúdo de todas as etapas.
- Formatação do arquivo final.