Atividade Prática 2 — Técnicas de Programação (10.0 pontos)

Prof. Me. Fernando Esquírio & Prof. Me. Matheus Fernandes

Regras:

- A atividade ficará aberta no Blackboard até a data limite de entrega;
- A atividade deverá ser entregue exclusivamente pelo Blackboard;
- Prazo de entrega: 04/11/2020 até as 23:59h
- Os alunos que não tiverem acesso ao Blackboard deverão entrar em contato com o professor;
- A submissão da atividade deve ser feita em um único arquivo zip. Este arquivo deve ter o nome do aluno, ex: Joao da Silva.zip. Este zip deve conter apenas os arquivos fonte e os cabeçalhos implementadas para a solução dos exercícios muito bem identificados;
- Caso seu código não compile ou apresente erros de execução (crash, loop infinito, etc.), a nota daquele exercício automaticamente é zerada;
- Códigos entregues em qualquer outra linguagem que não seja C não serão corrigidos e serão zerados. Exemplos de algumas linguagens não permitidas: Java, C++, C#, Python, JavaScript, GO, Ruby, etc.
- A atividade deve ser feita de maneira individual ou em dupla. Para o trabalho feito em dupla, cada aluno deverá postar a sua atividade no seu portal e aos alunos deverão criar um comentário com os nomes dos alunos e RA no início do arquivo fonte com a função main.
- Caso dois alunos/duplas submetam códigos iguais ou muito semelhantes, será considerado plágio. Consequentemente, ambos os/as alunos/duplas receberão zero na atividade inteira;
- A correção do código será feita de acordo com os critérios de correção listados abaixo.
- Siga cuidadosamente as instruções de cada exercício, atentando-se para corresponder com exatidão os requisitos de cada exercício;
- Teste seu código várias vezes com várias entradas diferentes, para ter certeza de que está atendendo a todos os critérios acima;
- O seu código será avaliado não apenas no quesito funcionamento, mas também a organização, nomenclatura de variáveis, boas práticas e qualidade dos comentários.
- Estamos fazendo uma avaliação com prazo fixo de entrega, entenda que não será possível sanar dúvidas extras por e-mail ou não haverá prorrogação de prazo.
- Caso o aluno/grupo faça mais de quatro upgrades, serão corrigidos os quatro primeiros que aparecem no arquivo fonte da função main(). O restante será desconsiderado.

Erros comuns:

- Não use acentuação ou sinais gráficos, tanto no código quanto nos nomes dos arquivos;
- Não importe a biblioteca <locale.h> e nem a <conio.h>;
- Não retorne outro valor além de 0 na função main;
- Verifique se está inicializando corretamente todas as variáveis, principalmente os acumuladores e contadores.
- Criar upgrades muito iguais ou muito semelhantes. Caso isso aconteça será considerado como um upgrade apenas.

Critérios de avaliação dos upgrades nos códigos:

- 0%: Upgrade não é válido ou sem sentido ou funciona para situações bem específicas.
- 50%: Upgrade não foi completo e funciona satisfatoriamente e apresenta falhas em algumas situações ou upgrade completo sem comentários explicativos ou com a explicação muito simples.
- 100%: Upgrade funciona corretamente sem erros nos testes e com a explicação completa do que foi realizado.

Não será considerado upgrades válidos:

- Alterar variáveis onde os dados serão armazenados, por exemplo: trocar struct para vetores ou matrizes.
- Alterar forma de leitura dos dados, por exemplo: usar scanf ao invés de gets ou vice-versa, ler as variáveis de uma vez só ao invés de leituras individuais ou vice-versa.
- Alterar ordem de inserção ou exibição dos dados.
- Mudança de nome ou tipo de dados de variáveis ou de variável simples para vetores ou matrizes.
- Qualquer solução muito trivial, simples, que não seja significante no código.
- Inclusão de biblioteca padrão do C com a diretiva include.
- Função do tipo fflush, system, etc.

Importante:

- Use comentários apenas para colocar os nomes dos alunos e explicar os upgrades realizados.
- Exibição das informações no console. Formatar a exibição do menu e dos dados listados cadastrados. Um ou mais alterações só contará como um upgrade.

Referência muito útil:

- Site com a documentação das bibliotecas em C: http://www.cplusplus.com/reference/clibrary/

Exercícios 01 (5 pontos) – Empresa Sirius Cybernetics Corp.

A empresa Sirius Cybernetics Corp. tem 5 funcionários e deseja fazer construir um programa para inserir e exibir os dados desses funcionários. Um estagiário da empresa fez o seguinte programa básico para inserção e exibição de um funcionário.

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct func
{
  int id;
  char nome[50];
  long cpf;
  int dia;
  int mes;
  int ano;
} tFuncionario;

int main(){
  tFuncionario dados;
  dados.id = 1;
  //Leitura dos dados
  printf("\nDigite o nome do funcionario: ");
  gets(dados.nome);
  printf("\nDigite o CPF: ");
  scanf("%li", &dados.cpf);
  printf("\nDigite a data de nascimento: ");
  scanf("%i%i%i", &dados.dia, &dados.mes, &dados.ano);
  //Imprime os dados
  printf("\nID: %d\n", dados.id);
  printf("\nNome: %s\n", dados.nome);
  printf("\nCPF: %d\n", dados.cpf);
  printf("\nData de nascimento : %i/%i/%i\n\n", dados.dia, dados.mes, dados.ano);
  return 0;
}//Fim da main()
```

A empresa Sirius Cybernetics Corp. entrou em contato com você para realizar quatro upgrades necessários (Valor: 1,25 pontos cada upgrade) no código, sendo que dois upgrades são obrigatórios e dois upgrades são livres que os programadores podem sugerir.

Upgrades obrigatórios:

1. (Valor 1,25 pontos) Criar um menu inicial usando laços de repetição com as três opções: Primeira opção é inserir os dados dos funcionários da empresa, a Segunda opção é exibir os funcionários já cadastrados e a Terceira opção é sair do programa.
   (A). Requisitos obrigatórios:
      I. A primeira opção (Inserir dados) deve permitir inserir os dados de 1 até 5 funcionários, pois deverá ter uma condição de parada caso o usuário deseja inserir menos de 5 dados e não permitir que mais de 5 funcionários possam ser inseridos.
      II. A segunda opção deve prever o envio de uma mensagem de erro, caso o usuário tente listar os funcionários, mas não foi inserido nenhum conteúdo ainda.
   (B). Critérios de avaliação:
      a) 0%: não criou um menu com as opções e os requisitos solicitados ou colocou um código sem sentido. Códigos semelhantes e comentários semelhantes entre trabalhos entregues, por exemplo, trocar nomes de variáveis e/ou trocar as palavras na explicação por sinônimos.

b) 50%: menu funciona perfeitamente, mas não possui comentários explicativos satisfatórios da lógica executada ou menu não atende um dos requisitos listados em (A).
c) 100%: menu funciona perfeitamente e possui comentários completos sobre a lógica implementada.

2. (Valor 1,25 pontos) Validação do nome do funcionário no momento de inserção de dados do funcionário.
   (A). Requisitos obrigatórios:
      I. Verificar se o nome do funcionário inserido contém somente letras maiúsculas ou minúsculas do alfabeto inglês (sem acentuação gráfica e caracteres especial) e espaço em branco.
      II. Ignorar nomes maiores do que 49 caracteres.
   (B). Critérios de avaliação:
      a) 0%: não criou a validação com os requisitos solicitados ou colocou um código sem sentido. Códigos semelhantes e comentários semelhantes entre trabalhos entregues, por exemplo, trocar nomes de variáveis e/ou trocar as palavras na explicação por sinônimos.
      b) 50%: criou a validação perfeitamente, mas não possui comentários explicativos satisfatórios da lógica executada ou a validação não atende satisfatoriamente um dos requisitos listados em (A).
      c) 100%: menu funciona perfeitamente e possui comentários completos sobre a lógica implementada.

Sugestões de upgrades livres, mas vocês podem fazer algo que quiserem:

- ✓ Inserir um ou mais campos na struct e criar a validação para esses campos. A inserção de um ou mais campos só contará como um upgrade e não serão aceitos a inserção de campos sem a implementação da validação dele.
  - Exemplos:
  - Um campo salário anual e validar se o salário inserido está entre R$0,00 e R$1000000,00.
  - Separar campos nome e sobrenome.
- ✓ Criar uma máscara de leitura do CPF digitado pelo usuário e validar o mesmo (Os dois contará como um upgrade e não serão considerados se forem implementados sozinhos).
  - O que seria uma máscara para a leitura do CPF?
  - Por exemplo, o usuário digita o CPF com os pontos e o traço 000.000.000-00 e o seu código irá ler apenas os números e ignorar os pontos e o traço.
- ✓ Criar uma máscara de leitura da data de nascimento digitada pelo usuário e validar o mesmo (Os dois contará como um upgrade e não serão considerados se forem implementados sozinhos).
  - O que seria uma máscara para a leitura da data de nascimento?
  - Por exemplo, o usuário digita a data com as barras 11/02/2010 e o seu código irá ler apenas os números e ignorar os outros caracteres.
  - Validação da data pode ser: não aceitar datas menores do que 01/01/1950 e não aceitar data maiores que a data do dia.
- ✓ Na segunda opção (Lista funcionários), permitir a busca dos dados do funcionário de acordo como a ID ou primeiro nome do funcionário.
- ✓ Exibição das informações no console. Formatar a exibição do menu e dos dados listados cadastrados. Um ou mais alterações só contará como um upgrade.
- ✓ Criar arquivos de cabeçalho .h e fonte .c de biblioteca com a implementação de pelo menos duas funções de verificação e/ou validação.

Os upgrades livres escolhidos pelos programadores serão avaliados em:

- 0%: Upgrade não é válido ou sem sentido ou funciona para situações bem específicas.
- 50%: Upgrade não foi completo e funciona satisfatoriamente e apresenta falhas em algumas situações ou upgrade completo sem comentários explicativos ou com a explicação muito simples.
- 100%: Upgrade funciona corretamente sem erros nos testes e com a explicação completa do que foi realizado.

Importante:

- Use comentários apenas para colocar os nomes dos alunos e explicar os upgrades realizados.
- Exibição das informações no console. Formatar a exibição do menu e dos dados listados cadastrados. Um ou mais alterações só contará como um upgrade.

Referência muito útil:

- Site com a documentação das bibliotecas em C: http://www.cplusplus.com/reference/clibrary/

Executáveis para vocês verem:

- Exercício 01 básico: https://Ex01Basico.mcf1110.repl.run
- Exercício 01 com upgrades obrigatórios e validação de CPF e Data: https://Ex01Obrigatorio.mcf1110.repl.run

Exercícios 02 (5 pontos) – Empresa do Malvado Doofenshmirtz

A Empresa do Malvado Doofenshmirtz quer fazer um código para receber um texto com até 200 caracteres do alfabeto inglês (letras maiúsculas e minúsculas, sem acentuação gráfica e caracteres especiais) e retorne a quantidade que cada letra do alfabeto aparece nesse texto. Você foi contratado para fazer o código e realizar quatro upgrades necessários com melhorias (Valor: 1 ponto cada upgrade) no programa, sendo dois obrigatórios e dois livre de sua escolha. Cuidado upgrades iguais ou muito semelhantes serão considerados como um upgrade.

1. Critérios de avaliação:
   (A). Código funcionando de forma simples (Diferenciando letras maiúsculas de minúsculas), recebendo o texto sem validações e verificações (Valor 1 pontos).
   (B). Quatro upgrades implementado (Valor 1 ponto cada upgrade).

Upgrades obrigatórios:

1. Não diferenciar letras maiúsculas e minúsculas, assim contabilizado em contadores distintos (Valor 1 ponto).
2. Ignorar qualquer Enter ou \n no texto digitado pelo usuário (1 ponto). Para finalizar a digitação, escolha outro critério (Dica: use a função fgetc(stdin)).

Os upgrades serão avaliados em:

- 0%: Upgrade não é válido ou sem sentido ou funciona para situações bem específicas.
- 50%: Upgrade não foi completo e funciona satisfatoriamente e apresenta falhas em algumas situações ou upgrade completo sem comentários explicativos ou com a explicação muito simples.
- 100%: Upgrade funciona corretamente sem erros nos testes e com a explicação completa do que foi realizado.

Importante:

- Use comentários apenas para colocar os nomes dos alunos e explicar os upgrades realizados.
- Exibição das informações no console. Formatar a exibição do menu e dos dados listados cadastrados. Um ou mais alterações só contará como um upgrade.

Referência muito útil:

- Site com a documentação das bibliotecas em C: http://www.cplusplus.com/reference/clibrary/

Executáveis para vocês verem:

- Exercício 02 básico: https://Ex02Basico.mcf1110.repl.run
- Exercício 02 com upgrades obrigatórios: https://Ex02Obrigatorio.mcf1110.repl.run

Daqui pra baixo não faz parte da avaliação e sim um exemplo do que será avaliado

Vamos mostrar um exemplo de avaliação de upgrade. Não faz parte da prova. Suponha um programa que recebe um número inteiro e imprime se ele é par ou ímpar.

Exemplo de **Códigos sem upgrade ou com upgrade sem mudança significativa (valerá 0% da nota)**

Código 01

```
#include <stdio.h>
int main(){
  int num = 0;
  printf("Digite um numero: ");
  scanf("%d", &num);
  if (num%2 == 0)
    printf("O numero %d eh par.\n", num);
  else
    printf("O numero %d eh impar.\n", num);
  return 0;
}
```

Código 02

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  int num = 0;
  printf("Digite um numero: ");
  scanf("%i", &num);
  fflush(stdin);
  if (!(num%2))
    printf("O numero %d eh par.\n", num);
  else
    printf("O numero %d eh impar.\n", num);
  return 0;
}
```

Exemplo de **Códigos com upgrade, mas não completo (sem comentários ou comentários simples) ou não funcionando corretamente (valerá 50% da nota)**

Código 03

```
#include <stdio.h>
int main(){
  int num = 0, estado = 0;
  //Testa se foi digitado um numero
  do{
    printf("Digite um numero: ");
    estado = scanf("%d", &num);
    if(estado == 0)
       printf("Valor invalido! \n\n");
  }while(estado == 0);

  if (!(num%2))
    printf("O numero %d eh par.\n", num);
  else
    printf("O numero %d eh impar.\n", num);
  return 0;
}
```

Código 04

```
#include <stdio.h>
int main(){
  int num = 0, estado = 0;
  char cont = 's';
  //Repetir a operacao ate o usuario nao quiser mais
  while(cont != 'n'){
    //Testa se foi digitado um numero
    do{
      printf("Digite um numero: ");
      estado = scanf("%d", &num);
      if(estado == 0)
        printf("Valor invalido! \n\n");
    }while(estado == 0);

    if (!(num%2))
      printf("O numero %d eh par.\n", num);
    else
      printf("O numero %d eh impar.\n", num);
    printf("Deseja continuar? s-sim n-nao");
    scanf(" %c", &cont);
  }
  return 0;
}
```

Exemplo **Códigos com upgrade e com a explicação completa da feature nova (100% da nota).**

Código 05	Código 06

#include <stdio.h>
int main(){
  int num = 0, estado = 0;
  /*Testa ser o valor digitado é um número de acordo
  com o valor retornado pela função scanf. A função
  scanf retorna a quantidade de itens efetivamente
  lidos. Dessa forma, se o valor o código de formatação
  é %d, a função retorna 0 0 para qualquer valor digitado
  que não seja um número e diferente de 0 para um
número.
  */
  do{
  printf("Digite um numero: ");
  estado = scanf("%d %d", &num, &num);
  printf("Estado: %d \n\n", estado);
  fflush(stdin);
  if(estado == 0)
  printf("Valor invalido! \n\n");
  }while(estado == 0);
 	
  if (!(num%2))
  printf("O numero %d eh par.\n", num);
  else
  printf("O numero %d eh impar.\n", num);
  return 0;
}
	#include <stdio.h>
	int main(){
	  int num = 0, estado = 0;
	  char cont = 's';
	  /*Repete o loop enquanto não for digitado o caractere n.
	  Para qualquer outro caractere digitado diferente de n
	  o laço continua, não necessariamente deve ser s.
	  */
	  while(cont != 'n'){
	  /*Testa ser o valor digitado é um número de acordo
	  com o valor retornado pela função scanf. A função
	  scanf retorna a quantidade de itens efetivamente
	  lidos. Dessa forma, se o valor o código de formatação
	  é %d, a função retorna 0 0 para qualquer valor digitado
	  que não seja um número e diferente de 0 para um número.
	  */
	  do{
	  printf("Digite um numero: ");
	  estado = scanf("%d %d", &num, &num);
	  printf("Estado: %d \n\n", estado);
	  fflush(stdin);
	  if(estado == 0)
	  printf("Valor invalido! \n\n");
	  }while(estado == 0);
	
	  if (!(num%2))
	  printf("O numero %d eh par.\n", num);
	  else
	  printf("O numero %d eh impar.\n", num);
	  printf("Deseja continuar? s-sim n-nao");
	  scanf(" %c", &cont);
	  }
	  return 0;
	}