UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

TRABALHO PRÁTICO 2

Aproveitando a onda da Inteligência Artificial e dos Modelos de Linguagens, Joãozinho teve a ideia de criar um ChatBot, um programa capaz de conversar com o usuário, fazer perguntas e dar respostas como se fosse uma pessoa real. Como ele ainda não fez a disciplina de IA e não sabe criar uma LLM, começou com um programa que interage com o usuário, mas sempre dando respostas prontas. Ajude Joãozinho a desenvolver esse ChatBot.

A tela abaixo mostra um exemplo de como o programa deve funcionar. O texto digitado pelo usuário é exibido na cor branca. Em amarelo a confirmação do programa, que sempre repete o texto lido. Em vermelho, a resposta dada pelo programa.

```
> Oi
> Entendi: "Oi"
> Como vai você?
> Bem, obrigado.
> Entendi: "Bem, obrigado."
> O que você faz?
> Ciência da Computação
> Entendi: "Ciência da Computação"
> É dificil programar?
> Nada, é *fácil*.
> Entendi: "Nada, é fácil."
> Você conhece Joãozinho?
> Claro, meu *ídolo*.
> Entendi: "Claro, meu ídolo."
> Ele me criou, sabe?
> Sério? O que é você?
> Entendi: "Sério? O que é você?"
> Sou um chatbot.
> *Uau*!
> Entendi: "Uau!"
> Tenho que ir! Tchau!
> %
```

As respostas devem ser guardadas em um vetor de ponteiros para constantes strings. A quantidade de respostas deve ser maior ou igual a 5 e o texto das respostas é de escolha livre.

Observe que o caractere * é um caractere especial que deve ser ignorado pelo ChatBot quando ele mostra o texto que entendeu.

```
> É dificil programar?
> Nada, é *fácil*.
> Entendi: "Nada, é fácil."
```

A conversa é finalizada quando o usuário fornece um caractere % como resposta.

Uma vez finalizada a conversa, o programa deve exibir o resumo das interações com o usuário, exibindo os textos entre asteriscos com um destaque, como no exemplo abaixo.

IMPLEMENTAÇÃO

Utilize o programa principal fornecido abaixo sem fazer nenhuma alteração. Implemente as funções e bibliotecas necessárias para que o programa produza o resultado esperado.

```
#include <iostream>
#include "Text.h"
#include "Bot.h"
using namespace std;

int main()
{
   Text text, line;
   Create(&text, 20);
   Create(&line, 100);

   int i = 0;
   Read(&line);
   while (CharAt(&line, 0) != '%')
   {
      Add(&text, &line);
      Print(&line);
      Answer(i++);
      Read(&line);
   }

   Text title;
   Init(&title, "Resumo da Conversa ");
   Boxed(&title);
   Highlight(&text);

   Destroy(&line);
   Destroy(&text);
```

FUNÇÕES E BIBLIOTECAS

O programa deve implementar as seguintes bibliotecas e funções:

Biblioteca Text

- Define um registro **Text** formado por: um ponteiro que aponta para um vetor de caracteres, o tamanho do texto contido no vetor e a capacidade máxima do vetor de caracteres.

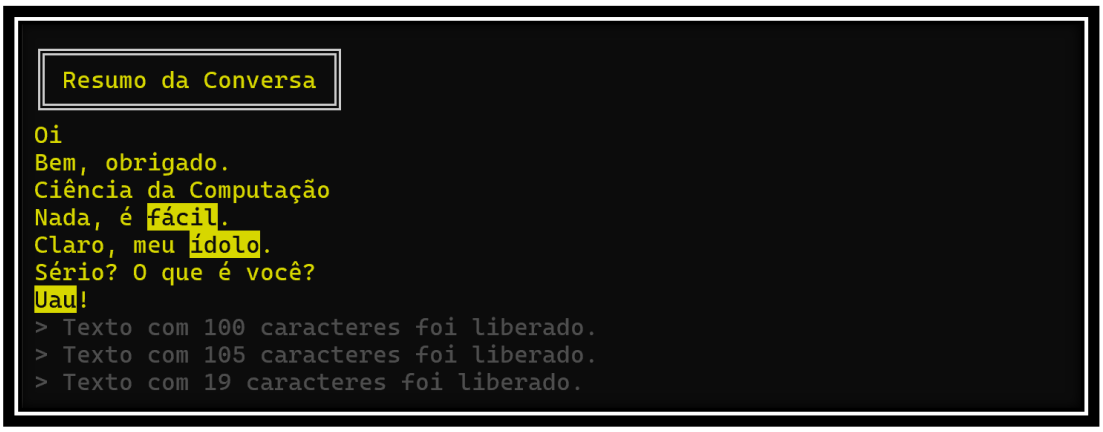
- **Create**: função recebe o endereço de um Text e um valor inteiro para definir o tamanho do vetor de caracteres. Não retorna valor. Ela deve alocar o vetor dinamicamente e inicializar apropriadamente os demais membros do registro.

  A função deve usar as diretivas de pré-processamento #ifdef e #endif para exibir uma mensagem com a quantidade de caracteres alocados quando o programa estiver sendo executado em modo DEBUG.

- **Init**: função recebe o endereço de um Text e uma constante string. Não retorna valor. Ela deve criar um vetor dinâmico de caracteres com o tamanho igual ao número de caracteres da constante string, sem contar o '\0'. Os membros tamanho e capacidade do registro Text devem ser ajustados de acordo com a string recebida. A função deve usar um laço para copiar os caracteres da constante string para o vetor dinâmico, sem o '\0'.

- **Destroy**: função recebe o endereço de um Text e libera a memória previamente alocada cujo endereço se encontra no seu primeiro membro.

  A função deve usar as diretivas de pré-processamento #ifdef e #endif para exibir uma mensagem com a quantidade de caracteres liberados quando o programa estiver sendo executado em modo DEBUG.



- **Show**: função recebe o endereço de um Text. Não retorna nada. Ela utiliza um laço for para exibir todos os caracteres do texto, ignorando os caracteres especiais * e o caractere '\n' presente no final do texto.

- **Print**: função recebe o endereço de um Text. Não retorna nada. Ela modifica a cor do texto para amarelo e chama a função Show para mostrar o texto. Observe que o texto deve ser exibido entre aspas duplas depois de > Entendi:.

```
> O que você faz?
> Ciência da Computação
> Entendi: "Ciência da Computação"
```

- **Read**: função recebe o endereço de um Text. Não retorna nada. Ela utiliza um laço de repetição para ler o texto digitado pelo usuário caractere a caractere. A função deve encerrar a leitura quando ler o caractere '\n' ou atingir a capacidade máxima de caracteres do registro, o que acontecer primeiro. O caractere '\n' deve ser inserido ao final do texto lido. A função deve atualizar o registro com a quantidade de caracteres armazenados no vetor, incluindo o '\n'.

- **CharAt**: função recebe o endereço de um Text e uma posição dentro do texto. Ela retorna um caractere. A função deve verificar se a posição é válida e retornar o caractere contido naquela posição. Caso a posição seja inválida, o caractere nulo ('\0') deve ser retornado.

- **Grow**: função recebe o endereço de um Text e um novo tamanho para o texto. Ela deve criar um vetor dinâmico de caracteres com o tamanho recebido. Em seguida o conteúdo do vetor antigo deve ser copiado para o novo usando um laço. O vetor dinâmico antigo deve ser liberado da memória e os membros do registro devem ser atualizados para o novo vetor.

  A função deve usar as diretivas de pré-processamento #ifdef e #endif para exibir uma mensagem indicando a capacidade antiga e a nova capacidade do texto quando o programa estiver sendo executado em modo DEBUG.

```
> Nada, é *fácil*.
> Adicionando "Nada, é fácil."
> Expandindo texto de 40 para 57 caracteres.
> Entendi: "Nada, é fácil."
> Você conhece Joãozinho?
```

- **Add**: função recebe dois endereços de elementos do tipo Text, sendo o primeiro o destino e o segundo a fonte. Ela não retorna nada. A função deve adicionar o texto contido na fonte ao final do texto armazenado no destino. Caso não tenha espaço suficiente no destino, a função **Grow** deve ser chamada para expandir a capacidade do destino. A nova capacidade deve ser exatamente igual a soma dos tamanhos dos textos da fonte e destino.

  A função deve usar as diretivas de pré-processamento #ifdef e #endif para exibir uma mensagem indicando o texto que está sendo adicionado quando o programa estiver sendo executado em modo DEBUG.

- **Hightlight**: função recebe o endereço de um Text. Não retorna nada. Ela exibe o texto armazenado no registro, destacando com cores invertidas o texto que estiver cercado

pelos caracteres especiais *. Utilize códigos ANSI para colorir o texto e o fundo do texto de forma a gerar o efeito abaixo.

- **Boxed**: função recebe o endereço de um Text. Não retorna nada. Ela deve exibir um adorno gráfico em formato de caixa ao redor do texto, como no exemplo abaixo. A caixa deve se adaptar ao tamanho do texto. Utilize laços para desenhar a caixa. A função Boxed deve utilizar a função **Hightlight** para mostrar o texto em si.

Biblioteca Bot

- Define um vetor de ponteiros para constantes strings com as respostas do robô. O texto das respostas é livre e a quantidade de respostas deve ser maior ou igual a 5.

- **Answer**: função recebe um inteiro e não retorna nada. Ela deve exibir a resposta contida na posição indicada pelo seu parâmetro. Se a posição ultrapassar o número de respostas do vetor, circule ao redor do vetor para pegar sempre uma resposta em uma posição válida.

INSTRUÇÕES GERAIS

- Utilize apenas conteúdos que foram vistos no curso, com as seguintes exceções:
  - É permitido usar a instrução condicional if-else.
  - É permitido usar operadores lógicos nos testes dos condicionais e dos laços.
- Não utilize variáveis globais.
- Use comentários, indentação e deixe o código organizado.

ENTREGA DO TRABALHO

Grupos: Trabalho individual

Data da entrega: 17/11/2025 (até a meia noite)

Valor do Trabalho: 3,0 pontos (na 2a Unidade)

Forma de entrega:

Ativar a exibição de arquivos ocultos no explorador de arquivos, apagar a pasta oculta .vs da pasta da solução, apagar a pasta x64 da pasta da solução, apagar a pasta x64 da pasta do projeto, apagar arquivo com a extensão .suo, se ele existir em qualquer uma das pastas.

Após as exclusões, confira se os arquivos restantes, que estão na pasta da solução e na pasta do projeto, têm uma das extensões a seguir. Nenhum outro tipo de arquivo deve ser enviado.

- .h – arquivo de cabeçalho
- .cpp – arquivo de código fonte
- .sln – solução do visual studio
- .slnx – solução do visual studio
- .vcxproj – projeto do visual studio
- .vcxproj.filters – filtros do projeto do visual studio
- .vcxproj.user – configurações do projeto do visual studio

Em seguida compactar todo o conteúdo da pasta da solução em um **arquivo .zip** (não usar .rar, .7zip ou outros formatos de compressão) e enviar através da tarefa correspondente no SIGAA.

Penalidades:

O não cumprimento das orientações acima resultarão em descontos na pontuação do trabalho:

- Programa não executa no Visual Studio 2022 (3,0 pontos)
- Programa contém partes de outros trabalhos (3,0 pontos)
- Programa utiliza recursos não vistos na disciplina (3,0 pontos)
- Atraso na entrega (1,5 pontos por dia de atraso)
- Arquivo compactado em outro formato que não zip (0,5 ponto)
- Envio de outros arquivos ou pastas que não sejam os descritos acima (0,5 ponto)
- Programa sem comentários (0,5 ponto)
- Código sem indentação e/ou desorganizado (0,5 ponto)
- Nomes de variáveis e funções sem significado (0,5 ponto)