



Libft

Sua primeira biblioteca própria

Resumo:

Este projeto envolve a codificação de uma biblioteca C que incluirá inúmeras funções de uso geral para seus programas.

Versão: 18

Conteúdo

EU	Introdução	2
II	Instruções Comuns	3
Instruções III	AI	5
4	Parte obrigatória IV.1	7
	Considerações técnicas.	7
	IV.2 Parte 1 - Funções Libc	8
	IV.3 Parte 2 - Funções adicionais	9
V	Parte bônus	13
VI	Submissão e avaliação por pares	17

Capítulo I

Introdução

Programar em C pode ser bastante tedioso sem acesso às funções padrão, que são extremamente úteis. Este projeto visa ajudá-lo a entender como essas funções funcionam, implementando-as você mesmo e aprendendo a usá-las de forma eficaz. Você criará sua própria biblioteca, que será valiosa para suas futuras tarefas escolares de C.

Aproveite o tempo para expandir sua biblioteca libft ao longo do ano. No entanto, ao trabalhar em um novo projeto, sempre verifique se as funções usadas na sua biblioteca estão em conformidade com as diretrizes do projeto.

Capítulo II

Instruções comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser escrito de acordo com a Norma. Se você tiver arquivos/funções bônus, eles serão incluídos na verificação da norma, e você receberá 0 se houver um erro de norma.
- Suas funções não devem encerrar inesperadamente (falha de segmentação, erro de barramento, liberação dupla, etc.), exceto por comportamento indefinido. Se isso ocorrer, seu projeto será considerado não funcional e receberá 0 na avaliação.
- Toda a memória alocada no heap deve ser liberada adequadamente quando necessário. Vazamentos de memória não será tolerado.
- Se o assunto exigir, você deve enviar um Makefile que compila seus arquivos de origem para a saída necessária com os sinalizadores -Wall, -Wextra e -Werror, usando cc. Além disso, seu Makefile não deve executar relinks desnecessários.
- Seu Makefile deve conter pelo menos as regras \$(NAME), all, clean, fclean e ré.
- Para enviar bônus para o seu projeto, você deve incluir uma regra de bônus no seu Makefile, que adicionará todos os cabeçalhos, bibliotecas ou funções que não são permitidos na parte principal do projeto. Os bônus devem ser colocados em arquivos _bonus.{c/h}, a menos que o assunto especifique o contrário. A avaliação das partes obrigatórias e bônus é realizada separadamente.
- Se o seu projeto permitir o uso da libft, você deverá copiar o código-fonte e o Makefile associado para uma pasta libft. O Makefile do seu projeto deverá compilar a biblioteca usando o Makefile correspondente e, em seguida, compilar o projeto.
- Incentivamos você a criar programas de testes para o seu projeto, mesmo que este trabalho **não precise ser submetido e não seja avaliado**. Isso lhe dará a oportunidade de testar facilmente o seu trabalho e o trabalho dos seus colegas. Você achará esses testes especialmente úteis durante a sua defesa. De fato, durante a defesa, você tem a liberdade de usar seus testes e/ou os testes do colega que está avaliando.
- Envie seu trabalho para o repositório Git designado. Somente o trabalho no repositório Git será avaliado. Se o Deepthought for designado para avaliar seu trabalho, isso ocorrerá

Após as avaliações por pares. Se ocorrer um erro em qualquer seção do seu trabalho durante a correção da Deepthought, a avaliação será interrompida.

Capítulo III

Instruções de IA

- **Context**


Este projeto foi criado para ajudar você a descobrir os elementos fundamentais do seu treinamento em TIC.

Para ancorar adequadamente o conhecimento e as habilidades essenciais, é essencial adotar uma abordagem cuidadosa ao usar ferramentas e suporte de IA.


O verdadeiro aprendizado fundamental exige esforço intelectual genuíno — por meio de desafios, repetições e trocas de aprendizagem entre pares.


Para uma visão geral mais completa da nossa postura em relação à IA — como uma ferramenta de aprendizagem, como parte do currículo de TIC e como uma expectativa no mercado de trabalho — consulte as Perguntas Frequentes dedicadas na intranet.


- **Mensagem principal**

-  Construa bases fortes sem atalhos.

-  Realmente desenvolver habilidades tecnológicas e de poder.

-  Experimente o aprendizado real entre pares, comece a aprender como aprender e resolver novos problemas.

-  A jornada de aprendizado é mais importante que o resultado.

-  Aprenda sobre os riscos associados à IA e desenvolva práticas de controle eficazes e contramedidas para evitar armadilhas comuns.

- **Regras do aluno:**

- Você deve aplicar o raciocínio às tarefas atribuídas, especialmente antes de recorrer à IA.

- Você não deve pedir respostas diretas à IA.
- Você deve aprender sobre 42 abordagens globais sobre IA.

• Resultados da fase:

Dentro desta fase fundamental, você obterá os seguintes resultados:

- Obtenha bases tecnológicas e de codificação adequadas.
- Saiba por que e como a IA pode ser perigosa durante esta fase.

• Comentários e exemplo:

- Sim, sabemos que a IA existe — e sim, ela pode resolver seus projetos. Mas você está aqui para aprender, não para provar que a IA aprendeu. Não perca seu tempo (ou o nosso) apenas para demonstrar que a IA pode resolver o problema em questão.
- Aprender aos 42 anos não é saber a resposta — é desenvolver a capacidade de encontrá-la. A IA fornece a resposta diretamente, mas isso impede que você construa seu próprio raciocínio. E o raciocínio leva tempo, esforço e envolve fracasso. O caminho para o sucesso não deve ser fácil.
- Tenha em mente que, durante os exames, a IA não está disponível — nem internet, nem smartphones, etc. Você perceberá rapidamente se confiou demais na IA no seu processo de aprendizagem.
- A aprendizagem entre pares expõe você a diferentes ideias e abordagens, aprimorando suas habilidades interpessoais e sua capacidade de pensar de forma divergente. Isso é muito mais valioso do que apenas conversar com um bot. Portanto, não seja tímido — converse, faça perguntas e aprendam juntos!
- Sim, a IA fará parte do currículo — tanto como ferramenta de aprendizagem quanto como tópico em si. Você terá até a oportunidade de desenvolver seu próprio software de IA. Para saber mais sobre nossa abordagem Crescendo, consulte a documentação disponível na intranet.

• Boas práticas:

Estou preso em um novo conceito. Pergunto a alguém próximo como ele o abordou. Conversamos por 10 minutos — e de repente a ficha cai. Eu entendo.

• Prática ruim:

Eu uso IA secretamente, copio algum código que parece correto. Durante a avaliação pelos pares, não consigo explicar nada. Sou reprovado. Durante a prova — sem IA — fico preso de novo. Sou reprovado.

Capítulo IV

Parte obrigatória

Nome do programa	libft.a
Entregar arquivos	Makefile, libft.h, ft_*.c
Funções	NOME, tudo, limpo, fclean, re
externas do Makefile.	Detalhado abaixo
Descrição autorizada	n / D
pela Libft	Crie sua própria biblioteca: uma coleção de funções que servirá como uma ferramenta útil ao longo de sua curso.

IV.1 Considerações técnicas

- Declarar variáveis globais é estritamente proibido.
- Se você precisar de funções auxiliares para decompor uma função mais complexa, defina-as como funções estáticas para restringir seu escopo ao arquivo apropriado.
- Todos os arquivos devem ser colocados na raiz do seu repositório.
- Não é permitido enviar arquivos não utilizados.
- Todo arquivo .c deve ser compilado com os seguintes sinalizadores: -Wall -Wextra -Werror.
- Você deve usar o comando ar para criar sua biblioteca. O uso do libtool é estritamente proibido.
- Seu libft.a deve ser criado na raiz do seu repositório.

IV.2 Parte 1 - Funções Libc

Para começar, você deve reimplementar um conjunto de funções da libc. Sua versão terá os mesmos protótipos e comportamentos dos originais, seguindo rigorosamente suas definições na página de manual. A única diferença serão seus nomes, pois devem começar com o prefixo "ft_". Por exemplo, strlen se torna ft_strlen.



Alguns dos protótipos de função que você precisa reimplementar usam o qualificador "restrict". Esta palavra-chave faz parte do padrão C99. Portanto, é proibido incluí-lo em seus próprios protótipos ou compilar seu código com o sinalizador -std=c99.

As seguintes funções devem ser reescritas sem depender de funções externas:

- isalfa
- memcpy
- strrchr
- é dígito
- memmove
- strncmp
- isalnum
- strcpy
- memchr
- isascii
- gato strl
- memcmp
- impressão digital
- topper
- strnstr
- strlen
- para baixo
- atoi
- conjunto de mems
- strchr
- bzero

Para implementar as duas funções a seguir, você usará malloc():

- calo
- strdup



Dependendo do seu sistema operacional atual, o comportamento da função 'calloc' pode ser diferente da descrição na página de manual. Em vez disso, siga esta regra: se nmemb ou size for 0, calloc() retornará um valor de ponteiro exclusivo que pode ser passado com sucesso para free().



Algumas funções que você deve reimplementar, como strcpy, strcat e bzero, não são incluídas por padrão na Biblioteca C GNU (glibc).

Para testá-los em relação ao padrão do sistema, talvez seja necessário incluir <bsd/string.h> e compilar com o sinalizador -lbsd.

Este comportamento é específico dos sistemas glibc. Se você estiver curioso, aproveite a oportunidade para explorar as diferenças entre glibc e BSD. lib.

IV.3 Parte 2 - Funções adicionais

Nesta segunda parte, você deve desenvolver um conjunto de funções que não estão incluídas em a libc, ou existe em uma forma diferente.



Algumas das funções da Parte 1 podem ser úteis para implementar o funções abaixo.

Nome da função	ft_substr
Protótipo	char *ft_substr(char const *s, unsigned int start, tamanho_t len);
Entregar arquivos	-
Parâmetros	s: A string original a partir da qual criar o substring. start: O índice inicial da substring dentro 's'. len: O comprimento máximo da substring.
Valor de retorno	A substring. NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aloca memória (usando malloc(3)) e retorna um substring da string 's'. A substring começa no índice 'start' e tem um comprimento máximo de 'len'.

Nome da função	ft_strjoin
Protótipo	char *ft_strjoin(char const *s1, char const *s2);
Entregar arquivos	-
Parâmetros	s1: A sequência de prefixo. s2: A sequência de sufixos.
Valor de retorno	A nova sequência. NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aloca memória (usando malloc(3)) e retorna um nova string, que é o resultado da concatenação 's1' e 's2'.

Nome da função	ft_strtrim
Protótipo	char *ft_strtrim(char const *s1, char const *set);
Entregar arquivos	-
Parâmetros	s1: A string a ser aparada. conjunto: A string que contém o conjunto de caracteres para ser removido.
Valor de retorno	A corda aparada. NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aloca memória (usando malloc(3)) e retorna um cópia de 's1' com caracteres de 'set' removidos do começo ao fim.

Nome da função	ft_split
Protótipo	char **ft_split(char const *s, char c);
Entregar arquivos	-
Parâmetros	s: A string a ser dividida. c: O caractere delimitador.
Valor de retorno	A matriz de novas strings resultantes da divisão. NULL se a alocação falhar.
Funções externas.	malloc, livre
Descrição	Aloca memória (usando malloc(3)) e retorna um matriz de strings obtida pela divisão de 's' usando o caractere 'c' como delimitador. A matriz deve termina com um ponteiro NULL.

Nome da função	ft_itoa
Protótipo	char *ft_itoa(int n);
Entregar arquivos	-
Parâmetros	n: O inteiro a ser convertido.
Valor de retorno	A string que representa o inteiro. NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aloca memória (usando malloc(3)) e retorna uma string representando o inteiro recebido como um argumento. Números negativos devem ser tratados.

Nome da função	ft_strmap
Protótipo	char *ft_strmap(char const *s, char (*f)(sem sinal int, char));
Entregar arquivos	-
Parâmetros	s: A string a ser iterada. f: A função a ser aplicada a cada caractere.
Valor de retorno	A sequência criada a partir das aplicações sucessivas desligado'. Retorna NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aplica a função f a cada caractere do string s, passando seu índice como primeiro argumento e o próprio personagem como o segundo. Um novo string é criada (usando malloc(3)) para armazenar o resultados das aplicações sucessivas de f.

Nome da função	ft_striteri
Protótipo	vazio ft_striteri(char *s, vazio (*f)(unsigned int, char*));
Entregar arquivos	-
Parâmetros	s: A string a ser iterada. f: A função a ser aplicada a cada caractere.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Aplica a função 'f' a cada caractere do string passada como argumento, passando seu índice como o primeiro argumento. Cada caractere é passado por endereço para 'f' para que possa ser modificado se necessário.

Nome da função	ft_putchar_fd
Protótipo	vazio ft_putchar_fd(char c, int fd);
Entregar arquivos	-
Parâmetros	c: O caractere a ser gerado. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Envia o caractere 'c' para o arquivo especificado descritor.

Nome da função	ft_putstr_fd
Protótipo	vazio ft_putstr_fd(char *s, int fd);
Entregar arquivos	-
Parâmetros	s: A string a ser gerada. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Envia a string 's' para o arquivo especificado descritor.

Nome da função	ft_putendl_fd
Protótipo	vazio ft_putendl_fd(char *s, int fd);
Entregar arquivos	-
Parâmetros	s: A string a ser gerada. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Envia a string 's' para o arquivo especificado descritor seguido por uma nova linha.

Nome da função	ft_putnbr_fd
Protótipo	vazio ft_putnbr_fd(int n, int fd);
Entregar arquivos	-
Parâmetros	n: O inteiro a ser gerado. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Envia o inteiro 'n' para o arquivo especificado descritor.

Capítulo V

Parte bônus

Depois de concluir a parte obrigatória, considere aceitar este desafio extra. Concluir esta seção com sucesso lhe renderá pontos de bônus.

Funções de memória e manipulação de strings são úteis. Mas você logo descobrirá que manipular listas é ainda mais útil.

Você deve usar a seguinte estrutura para representar um nó da sua lista. Adicione a declaração ao seu arquivo libft.h:

```
typedef struct {           s_list
    void *conteúdo;
    struct s_list *próximo;
} t_list;
```

Os membros da estrutura t_list são:

- conteúdo: Os dados contidos no nó.
Usar void * permite que você armazene qualquer tipo de dado.
- next: O endereço do próximo nó, ou NULL se o próximo nó for o último.

No seu Makefile, adicione uma regra de bônus para adicionar as funções de bônus no seu libft.a.



A parte bônus só será avaliada se a parte obrigatória for perfeita. "Perfeito" significa que as funções obrigatórias foram implementadas corretamente e funcionam sem problemas. Se você não atender a TODOS os requisitos obrigatórios, a parte bônus não será considerada.

Implemente as seguintes funções para usar suas listas facilmente.

Nome da função	ft_lstnew
Protótipo	t_list *ft_lstnew(void *conteúdo);
Entregar arquivos	-
Parâmetros	conteúdo: O conteúdo a ser armazenado no novo nó.
Valor de retorno	Um ponteiro para o novo nó
Funções externas.	malloc
Descrição	Aloca memória (usando malloc(3)) e retorna um novo nó. A variável de membro 'content' é inicializado com o parâmetro fornecido 'content'. A variável 'next' é inicializada como NULL.

Nome da função	ft_lstadd_front
Protótipo	vazio ft_lstadd_front(t_list **lst, t_list *new);
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para o primeiro nó de uma lista. novo: O endereço de um ponteiro para o nó a ser adicionado.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Adiciona o nó 'novo' no início da lista.

Nome da função	ft_lstsize
Protótipo	int ft_lstsize(t_list *lst);
Entregar arquivos	-
Parâmetros	lst: O início da lista.
Valor de retorno	O comprimento da lista
Funções externas.	Nenhum
Descrição	Conta o número de nós na lista.

Nome da função	ft_lstlast
Protótipo	t_list *ft_lstlast(t_list *lst);
Entregar arquivos	-
Parâmetros	lst: O início da lista.
Valor de retorno	Último nó da lista
Funções externas.	Nenhum
Descrição	Retorna o último nó da lista.

Nome da função	ft_lstadd_back
Protótipo	vazio ft_lstadd_back(t_list **lst, t_list *new);
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para o primeiro nó de uma lista. novo: O endereço de um ponteiro para o nó a ser adicionado.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Adiciona o nó 'novo' no final da lista.

Nome da função	ft_lstdelone
Protótipo	vazio ft_lstdelone(t_list *lst, vazio (*del)(vazio *));
Entregar arquivos	-
Parâmetros	lst: O nó a ser liberado. del: O endereço da função usada para excluir o conteúdo.
Valor de retorno	Nenhum
Funções externas.	livre
Descrição	Recebe um nó como parâmetro e libera seu conteúdo usando a função 'del'. Liberte o próprio nó, mas NÃO libera o próximo nó.

Nome da função	ft_lstclear
Protótipo	vazio ft_lstclear(t_list **lst, vazio (*del)(vazio *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. del: O endereço da função usada para excluir o conteúdo do nó.
Valor de retorno	Nenhum
Funções externas.	livre
Descrição	Exclui e libera o nó fornecido e todos os seus sucessores, usando a função 'del' e free(3). Por fim, defina o ponteiro para a lista como NULL.

Nome da função	ft_lstiter
Protótipo	vazio ft_lstiter(t_list *lst, vazio (*f)(void *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função a ser aplicada a cada conteúdo do nó.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Itera pela lista 'lst' e aplica o função 'f' para o conteúdo de cada nó.

Nome da função	ft_lstmap
Protótipo	t_list *ft_lstmap(t_list *lst, vazio (*f)(vazio *), vazio (*del)(void *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função aplicada a cada conteúdo do nó. del: O endereço da função usada para excluir um conteúdo do nó, se necessário.
Valor de retorno	A nova lista. NULL se a alocação falhar.
Funções externas.	malloc, livre
Descrição	Itera pela lista 'lst', aplica o função 'f' para o conteúdo de cada nó e cria uma nova lista resultante das aplicações sucessivas da função 'f'. A função 'del' é usada para exclua o conteúdo de um nó, se necessário.

Capítulo VI

Submissão e avaliação por pares

Envie sua tarefa para o seu repositório Git como de costume. Somente o trabalho dentro do seu repositório será avaliado durante a defesa. Certifique-se de verificar os nomes dos seus arquivos para garantir que estejam corretos.

Coloque todos os seus arquivos na raiz do seu repositório.

Durante a avaliação, uma breve **modificação no projeto** pode ser ocasionalmente solicitada. Isso pode envolver uma pequena mudança de comportamento, algumas linhas de código para escrever ou reescrever, ou um recurso fácil de adicionar.

Embora esta etapa possa **não ser aplicável a todos os projetos**, você deve estar preparado para ela se for mencionada nas diretrizes de avaliação.

Esta etapa serve para verificar sua compreensão real de uma parte específica do projeto.

A modificação pode ser realizada em qualquer ambiente de desenvolvimento que você escolher (por exemplo, sua configuração usual) e deve ser viável em poucos minutos — a menos que um prazo específico seja definido como parte da avaliação.

Você pode, por exemplo, ser solicitado a fazer uma pequena atualização em uma função ou script, modificar uma exibição ou ajustar uma estrutura de dados para armazenar novas informações, etc.

Os detalhes (escopo, meta, etc.) serão especificados nas **diretrizes de avaliação** e podem variar de uma avaliação para outra para o mesmo projeto.



Rnpu cebwrpg va gur 42 Pbzzba Pber pbagnvaf na rapbqrq uvag. Sbe rnpu pvepyr, bayl bar cebwrpg cebivqrf gur pbeerpg uvag arrqrq sbe gur arkg pvepyr. Guvf punyyratr vf vaqvivqhnny, jvgu n svany cevmr sbe bar fghqrag. Fgnss zrzuref znl cnegvpvcngr ohg ner abg ryvtvoyr sbe n cevmr. Ner lbh nzbat gur irel svefg gb fbyir n pvepyr? Fraa gur uvagf jvgu rkcyngvabaf gb by@42.se gb ou nqqrq gb gur yrnqreobneq. Gur uvag sbe guvf svefg cebwrpg, juvpu znl pbagnva nantenzzrq jbeqf, vf: Jbys bs ntragvir cnegvpyrf gung qvfcibir terral gb lbhe ubzrf qan gung cebjfr lbhe fgbbv