Libft

Sua primeira biblioteca própria

Resumo:
Este projeto trata da codificação de uma biblioteca C.
Ele conterá muitas funções de uso geral das quais seus programas dependerão.

Versão: 15

Conteúdo

EU **Introdução** **2**

II Instruções Comuns **3**

III Parte obrigatória III.1 **5**

Considerações técnicas 5

III.2 Parte 1 - Funções Libc 6

III.3 Parte 2 - Funções adicionais 7

IV Parte Bônus **11**

V Submissão e avaliação por pares **15**

Capítulo I

Introdução

A programação em C pode ser muito tediosa quando não se tem acesso às funções padrão altamente úteis. Este projeto é sobre entender como essas funções funcionam, implementando e aprendendo a usá-las. Você criará sua própria biblioteca. Será útil, pois você o usará em suas próximas tarefas da escola C.

Aproveite o tempo para expandir sua liberdade ao longo do ano. No entanto, ao trabalhar em um novo projeto, não se esqueça de garantir que as funções usadas em sua biblioteca sejam permitidas nas diretrizes do projeto.

Capítulo II

Instruções Comuns

- Seu projeto deve ser escrito em C.
- Seu projeto deve ser redigido de acordo com a Norma. Se você tiver arquivos/funções de bônus, eles serão incluídos na verificação da norma e você receberá um 0 se houver um erro de norma dentro.
- Suas funções não devem ser encerradas inesperadamente (falha de segmentação, erro de barramento, double free, etc.) além de comportamentos indefinidos. Se isso acontecer, seu projeto será considerado não funcional e receberá nota 0 na avaliação.
- Todo o espaço de memória alocado no heap deve ser liberado adequadamente quando necessário. Sem vazamentos será tolerado.
- Se o assunto exigir, você deve enviar um Makefile que irá compilar seus arquivos de origem para a saída necessária com os sinalizadores -Wall, -Wextra e -Werror, use cc e seu Makefile não deve revincular.
- Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e ré.
- Para ativar bônus em seu projeto, você deve incluir uma regra de bônus em seu Makefile, que adicionará todos os vários cabeçalhos, bibliotecas ou funções que são proibidas na parte principal do projeto. Os bônus devem estar em um arquivo diferente _bonus.{c/h} se o assunto não especificar mais nada. A avaliação da parte obrigatória e bônus é feita separadamente.
- Se seu projeto permite que você use sua libft, você deve copiar suas fontes e seu Makefile associado em uma pasta libft com seu Makefile associado. O Makefile do seu projeto deve compilar a biblioteca usando seu Makefile e, em seguida, compilar o projeto.
- Incentivamos você a criar programas de teste para o seu projeto, mesmo que este trabalho **não precise ser enviado e não seja avaliado.** Isso lhe dará a chance de testar facilmente seu trabalho e o de seus colegas. Você achará esses testes especialmente úteis durante sua defesa. De fato, durante a defesa, você é livre para usar seus testes e/ou os testes do par que está avaliando.
- Envie seu trabalho para o repositório git atribuído. Somente o trabalho no repositório git será avaliado. Se o Deepthought for designado para avaliar seu trabalho, isso será feito

depois de suas avaliações de pares. Se ocorrer um erro em qualquer seção do seu trabalho durante a avaliação do Deepthought, a avaliação será interrompida.

Capítulo III

parte obrigatória

Nome do programa	libft.a
Entregar arquivos	Makefile, libft.h, ft_*.c NAME, all, clean,
Makefile	fclean, re Detalhado abaixo n/d Escreva
Funções externas.	sua própria biblioteca:
Libft autorizado	uma
Descrição	coleção de funções que será uma ferramenta útil para seu cursus.

III.1 Considerações técnicas

- É proibido declarar variáveis globais.
- Se você precisar de funções auxiliares para dividir uma função mais complexa, defina-as como estáticas funções. Dessa forma, seu escopo será limitado ao arquivo apropriado.
- Coloque todos os seus arquivos na raiz do seu repositório.
- É proibido entregar arquivos não utilizados.
- Todos os arquivos .c devem ser compilados com as flags -Wall -Wextra -Werror.
- Você deve usar o comando ar para criar sua biblioteca. Usando o comando libtool é proibido.
- Seu libft.a deve ser criado na raiz do seu repositório.

III.2 Parte 1 - Funções Libc

Para começar, você deve refazer um conjunto de funções da libc. Suas funções terão os mesmos protótipos e implementarão os mesmos comportamentos das originais. Eles devem obedecer à maneira como são definidos em seu homem. A única diferença serão seus nomes. Eles começarão com o prefixo 'ft_'. Por exemplo, strlen torna-se ft_strlen.

Alguns dos protótipos das funções que você precisa refazer usam o qualificador 'restringir'. Esta palavra-chave faz parte do padrão c99. Portanto, é proibido incluí-lo em seus próprios protótipos e compilar seu código com o sinalizador -std=c99.

Você deve escrever sua própria função implementando as seguintes funções originais. Eles não requerem nenhuma função externa:

- isalfa
- isdigito
- isalnum
- isascii
- isprint
- strlen
- memset
- bzero
- memcpy
- movimento de memória
- strlcpy
- strlcat
- superior
- abaixar
- strchr
- strrchr
- strncmp
- memchr
- memcmp
- strnstr
- atoi

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

- calloc
- strdup

III.3 Parte 2 - Funções adicionais

Nesta segunda parte, você deve desenvolver um conjunto de funções que não estão na libc ou que fazem parte dela, mas de uma forma diferente.

Algumas das seguintes funções podem ser úteis para escrever o funções da Parte 1.

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

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

Nome da função	ft_strtrim char
Protótipo	*ft_strtrim(char const *s1, char const *set);
Entregar arquivos	-
Parâmetros	s1: A string a ser aparada. set: O conjunto de referência de caracteres a serem cortados.
Valor de retorno	A corda aparada. NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aloca (com malloc(3)) e retorna uma cópia de 's1' com os caracteres especificados em 'set' removidos do início e do final da string.

Nome da função	ft_split char
Protótipo	**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, free Aloca
Descrição	(com malloc(3)) e retorna um array de strings obtido dividindo 's' usando o caractere 'c' como delimitador. A matriz deve terminar com um ponteiro NULL.

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

Nome da função	ft_strmapi char
Protótipo	*ft_strmapi(char const *s, char (*f)(unsigned int, char));
Entregar arquivos	-
Parâmetros	s: A string na qual iterar. f: A função a ser aplicada a cada caractere.
Valor de retorno	A string criada a partir das aplicações sucessivas de 'f'. Retorna NULL se a alocação falhar.
Funções externas.	malloc
Descrição	Aplica a função 'f' a cada caractere da string 's', passando seu índice como primeiro argumento para criar uma nova string (com malloc(3)) resultante de aplicações sucessivas de 'f'.

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

Nome da função	ft_putchar_fd void
Protótipo	ft_putchar_fd(char c, int fd);
Entregar arquivos	-
Parâmetros	c: O caractere para saída. fd: O descritor de arquivo no qual escrever.
Valor de retorno	Nenhum
Funções externas.	escrever
Descrição	Emite o caractere 'c' para o descritor de arquivo fornecido.

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

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

Nome da função	ft_putnbr_fd void
Protótipo	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	Emite o inteiro 'n' para o descritor de arquivo fornecido.

Capítulo IV

Parte bônus

Se você completou a parte obrigatória, não hesite em ir além fazendo esta parte extra. Ele trará pontos de bônus se for aprovado com sucesso.

Funções para manipular memória e strings são muito 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 sua declaração ao seu arquivo libft.h:

```
typedef struct {          s_list

  void                    *contente;
  struct s_list           *próximo;
                          lista_t;
```

Os membros da estrutura t_list são:

- conteúdo: Os dados contidos no nó. void * permite armazenar qualquer tipo de dados.
- próximo: O endereço do próximo nó, ou NULL se o próximo nó for o último.

Em seu Makefile, adicione uma regra de bônus para adicionar as funções de bônus ao seu libft.a.

A parte bônus só será avaliada se a parte obrigatória for PERFEITA. Perfeito significa que a peça obrigatória foi executada integralmente e funciona sem avarias. Se você não passou em TODOS os requisitos obrigatórios, sua parte de bônus não será avaliada.

Implemente as seguintes funções para usar facilmente suas listas.

Nome da função	ft_lstnew t_list
Protótipo	*ft_lstnew(void *conteúdo);
Entregar arquivos	-
Parâmetros	content: O conteúdo com o qual criar o nó.
Valor de retorno	O novo nó
Funções externas.	malloc
Descrição	Aloca (com malloc(3)) e retorna um novo nó. A variável de membro 'content' é inicializada com o valor do parâmetro 'conteúdo'. A variável 'next' é inicializada como NULL.

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

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

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

Nome da função	ft_lstdelone void
Protótipo	ft_lstdelone(t_list *lst, void (*del)(void *));
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	Toma como parâmetro um nó e libera a memória do conteúdo do nó usando a função 'del' fornecida como parâmetro e libera o nó. A memória de 'próximo' não deve ser liberada.

Nome da função	ft_lstclear void
Protótipo	ft_lstclear(t_list **lst, void (*del)(void *));
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 sucessores desse nó, usando a função 'del' e free(3). Por fim, o ponteiro para a lista deve ser definido como NULL.

Nome da função	ft_lstiter void
Protótipo	ft_lstiter(t_list *lst, void (*f)(void *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função usada para iterar em a lista.
Valor de retorno	Nenhum
Funções externas.	Nenhum
Descrição	Itera a lista 'lst' e aplica a função 'f' no conteúdo de cada nó.

Nome da função	ft_lstmap t_list
Protótipo	*ft_lstmap(t_list *lst, void *(*f)(void *), void (*del)(void *));
Entregar arquivos	-
Parâmetros	lst: O endereço de um ponteiro para um nó. f: O endereço da função usada para iterar em a lista. del: O endereço da função usada para excluir o conteúdo de um nó, se necessário.
Valor de retorno	A nova lista. NULL se a alocação falhar.
Funções externas.	malloc, free Itera a
Descrição	lista 'lst' e aplica a função 'f' no conteúdo de cada nó. Cria um novo lista resultante das sucessivas aplicações da função 'f'. A função 'del' é usada para exclua o conteúdo de um nó, se necessário.

Capítulo V

Submissão e avaliação por pares

Entregue sua atribuição em seu repositório Git como de costume. Apenas o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar novamente os nomes de seus arquivos para garantir que estejam corretos.

Coloque todos os seus arquivos na raiz do seu repositório.