Bacharelado em Ciência da Computação **DINF / UFPR**
CI067 - Oficina de Computação **1º semestre 2012**
Prof. Armando Luiz N. Delgado http://www.inf.ufpr.br/nicolui
TRABALHO 03 - TURMA C **25/5/2012**

Releases - Histórico:

r0	25/5/2012	Primeira versão
r1	30/5/2012	Pequenas alterações nas especificações das bibliotecas

1 Especificação do Trabalho

Criar um conjunto de bibliotecas de propósito geral que possam ser usados por qualquer programa.

1.1 Poliglota

Grupo Responsável: Alexandre Muller, Carlos Galves e Ruanito Santos

Esta biblioteca tem por objetivo fornecer toda a estrutura de dados e funções de manipulação que permita a um programa qualquer ter suas mensagens em diversos idiomas (português, inglês, mandarim, javanês arcaico, etc.).

Deve permitir as mudanças de idioma e dos textos das mensagens SEM A NECESSIDADE de recompilar e/ou alterar o código fonte da biblioteca.

A biblioteca deve permitir que a definição de mensagens seja por nome de programa, e para cada programa deve haver uma definição adequada das mensagens, agrupadas por idioma.

NO MÍNIMO, as seguintes funções devem estar presentes: definir/alterar idioma default, definir/alterar idioma corrente, imprimir o texto de um tipo de mensagem no idioma corrente, alterar um tipo de mensagem em um idioma associados a um programa, definir um novo tipo de mensagem e seu texto em um idioma associados a um programa, recuperar (sem imprimir) a lista de tipos de mensagens associados a um programa.

Quaisquer arquivos de dados que venham estar associados a esta biblioteca devem ser procurados nos diretórios login1-login2/lib/locale ou /home/especial/ci067/lib/locale. O idioma corrente inicialmente deve ser igual ao idioma default.

BÔNUS: Definir o idioma default das mensagens a partir do valor da variável de ambiente *shell* LANG.

Arquivos relacionados: libpoli.a, poli.c e poli.h

1.2 Listas

Grupo Responsável: Alan Peterson e Marcelo da Silveira

Esta biblioteca tem por objetivo fornecer toda a estrutura de dados e funções de manipulação que permita a um programa qualquer criar listas encadeada com elementos do tipo *float*, *int*, *string*. Defina um módulo de biblioteca com estruturas de dados adequadas, e, NO MÍNIMO, devem ser definidas funções para criar uma lista, inserir um elemento na lista, remover e buscar um elemento da lista identificado por uma chave, obter o valor armazenado em um nó da lista, obter o nó seguinte a outro nó da lista. As funções devem conter mecanismos próprios para um reuso econômico de memória.

BÔNUS: Definir a biblioteca de forma que a lista lide com tipos genéricos, isto é, deve ser possível criar as devidas estruturas com itens de dados de qualquer tipo. **DICA:** Uso de ponteiros para dados e ponteeiros para funções pode ajudar.

Arquivos relacionados: liblistas.a, listas.c e listas.h

1.3 Gerenciamento de Memória

Grupo Responsável: Lucas Affonso e Thiago Salvadori

Esta biblioteca tem por objetivo fornecer toda a estrutura de dados e funções de manipulação que permita a um programa qualquer alocar blocos de memória com controle interno de alocação, de forma que um bloco somente seja solicitado ao sistema operacional (via malloc, calloc ou realloc) se não houver blocos previamente alocados no programa mas que não estejam sendo usados.

NO MÍNIMO, as seguintes funções devem estar presentes: cialloc() (aloca e zera o conteúdo), cirealloc() (realoca bloco previamente alocado), cifree() (libera um bloco de memória, mas sem devolver para o sistema, mantendo disponível para novas alocações com cialloc() ou cirealloc()), cipurge() (libera todos os blocos de memória, devolvendo-os para o sistema, estejam eles ocupados ou não).

BÔNUS: fazer um mecanismo que, periodicamente, libere de volta para o sistema operacional os blocos de memória que não estejam sendo usados por um programa.

Arquivos relacionados: libgmemo.a, gmemo.c gmemo.h

2 Produto a ser Entregue

Cada biblioteca deve ser desenvolvida pelo grupo indicado na seção 1.

Cada grupo deve entregar um pacote de software completo contendo os fontes em linguagem C dos programas solicitados e documentação. O pacote deve ser arquivado e compactado com ***tar(1)*** e ***bzip2(1)*** em um arquivo chamado login1-login2.tar.bz2 (se grupo com 2 membros) ou login1-login2-login3.tar.bz2 (se grupo com 3 membros), onde login1, login2 e login3 são os *logins* dos alunos que compõem o grupo.

O pacote deve ter a seguinte estrutura de diretórios e arquivos:

- ./login1-login2/: diretório principal;
- ./login1-login2/LEIAME;
- ./login1-login2/src/: diretório contendo o arquivo Makefile e todos os arquivos fonte em linguagem C (*.c e *.h);
- ./login1-login2/lib/: diretório que conterá as bibliotecas (arquivos *.a), geradas após a compilação dos arquivos fontes da biblioteca;
- ./login1-login2/include/: diretório que conterá os cabeçalhos (arquivos *.h) das bibliotecas.

Note que a extração dos arquivos em login1-login2.tar.bz2 deve criar o diretório login1-login2 contendo todos os arquivos e diretórios acima.

2.1 Documentação

O pacote deve conter um arquivo de documentação em texto plano (ASCII). Este arquivo, chamado LEIAME, deve conter as seguintes informações:

- autoria do software, isto é, nome e RA dos membros do grupo;
- forma de uso da biblioteca por programas (descrição das funções, das estruturas de dados, de arquivos de configuração, se houver, chaves de compilação e includes);
- um capítulo especial descrevendo os algoritmos e as estruturas de dados utilizadas, as alternativas de implementação consideradas e/ou experimentadas e os motivos que o levaram a optar pela versão entregue, as dificuldades encontradas e as maneiras pelas quais foram contornadas.
- *bugs* conhecidos;
- outras informações que forem julgadas importantes.

2.2 Arquivos Fonte

Cada biblioteca deve ser implementada exclusivamente em linguagem C , e deve ser composto por um único arquivo de cabeçalho (extensão **.h**) contendo as definições de macros, tipos de dados, estruturas e protótipos de funções da biblioteca, e por 1 ou mais arquivos com extensão **.c** contendo implementação das diversas funções da biblioteca.

2.3 Arquivo Makefile

O arquivo Makefile deve possuir as regras necessárias para compilar os módulos individualmente e gerar o programa executável. As seguintes regras devem existir OBRIGATORIAMENTE:

tudo: gera as bibliotecas produzidas e as instala nos diretórios login1-login2/lib e login1-login2/include. Instalar significa copiar para os diretórios respectivos os arquivos *.a e *.h;

limpa: limpa os arquivos temporários como *.∼*, *.bak, core*;

faxina: limpa todos os arquivos temporários e os arquivos gerados pelo Makefile (*.o, executável em bin, etc.).

Modelos de Makefile podem ser encontrados aqui.

3 Entrega

A entrega do trabalho será feita em 3 (três) fases:

Fase 1: ENTREGA DA ESPECIFICAÇÃO (funções, arquivos, estruturas de dados) DA BIBLIOTECA DESENVOLVIDA NESTE TRABALHO EM **07 de junho de 2012**, IMPRETERIVELMENTE. Esta entrega deve constar do envio de um documento ASCII simples para o Prof. Armando Delgado <nicolui@inf.ufpr.br> com o Assunto (*Subject*): **CI067 - Espec. Trab 3**.

Fase 2: Apresentação do trabalho pelo grupo no dia 21 de junho de 2012em sala de aula para responder questionamentos do professor sobre o trabalho;

Fase 3: ENTREGA DO SOFTWARE COMPLETO DESENVOLVIDO NESTE TRABALHO EM **26 de junho de 2012, 22:00H**, IMPRETERIVELMENTE.

ATENÇÃO: O grupo que não cumprir qualquer uma das Fases terá nota 0 (zero). As datas são finais e não serão aceitos trabalhos entregues em atraso.

O produto do trabalho (seção 2) deve ser enviado como anexo por e-mail ao professor responsável pela disciplina:

- A mensagem com o anexo deve ser enviada ao Prof. Armando Delgado <nicolui@inf.ufpr.br> com o Assunto (*Subject*): **CI067 - Trab 3**.
- No corpo da mensagem DEVE CONSTAR OBRIGATORIAMENTE os Nomes e Números de Registro Acadêmico (RA) dos membros do grupo;
- O grupo deverá considerar o trabalho como entregue SOMENTE APÓS RECEBER DO PROFESSOR UMA MENSAGEM DE CONFIRMAÇÃO DE RECEBIMENTO dentro de 48 horas desde o envio da mensagem;

4 Critério de Avaliação

APENAS OS TRABALHOS QUE FUNCIONAREM SERÃO CORRIGIDOS. Se o trabalho não compilar ou acusar falha de segmentação (*Segmentation fault*) prematura durante os testes realizados pelo professor (sem que qualquer operação se efetue a contento), trará para o grupo NOTA 0 (ZERO). Também receberão NOTA 0 (ZERO) trabalhos plagiados de qualquer fonte, e/ou com códigos idênticos ou similares. Além disso, apenas trabalhos entregues no prazo marcado receberão nota.

Legibilidade, elegância, eficiência, modularidade, coesão e acoplamento entre módulos, e uso adequado de estruturas de dados serão levados em conta na avaliação.

Os algoritmos de manipulação das estruturas de dados (inserção, ordenação, etc.) devem ser tão eficientes quanto possível, usando todos os conceitos vistos nas disciplinas de Algoritmos do Curso. Ponteiros e alocação dinâmica de memória devem ser usados onde for eficiente e conveniente. Estruturas dinâmicas abordadas em Alg II, devidamente utilizadas, colaboram fortemente para uma avaliação positiva.

As manipulações dos itens de dados devem ser feitas em memória ou diretamente em disco, sempre levando em conta a eficiência do programa. Para implementação de conjuntos de itens usar a estruturação solicitada explicitamente na especificação (listas, vetores, filas, etc.). Caso não haja esta solicitação específica em algum caso, use a estruturação de dados mais eficiente possível.

Caso seu programa possua *bugs* conhecidos, descreva-os no arquivo LEIAME. A documentação de um *bug* pode evitar que o grupo receba nota Zero.

Os itens de avaliação do trabalho e respectivas pontuações são:

Qualidade da documentação: 15 pontos

Qualidade do código: 25 pontos

Implementação e Eficiência: 60 pontos

BÔNUS: Serão dados 25 pontos de BÔNUS ao grupo que implementar os ítens adicionais indicados em cada biblioteca da seção 1.

Defesa: A defesa do trabalho será oral, e definirá a nota individual de cada membro da equipe, de acordo com seu conhecimento a respeito do trabalho.

O item Qualidade da documentação se refere ao arquivo LEIAME. O item Qualidade do código inclui os comentários no código fonte e grau de portabilidade do código. O item Implementação e Eficiência inclui análise de estruturas de dados, de algoritmos utilizados e sua eficiência.

5 Casos Omissos

Quaisquer dúvidas a respeito da especificação, entrega ou avaliação do trabalho deverão ser encaminhadas aos professores da disciplina, pessoalmente ou através de mensagens eletrônicas.