ELC1067 - Laboratório de Programação II

Table of Contents

Professor: João Vicente Ferreira Lima.

Horário: segundas e quartas, 14h30.

Sala: 334.

Notas: link das avaliações

1 Informações

O objetivo da disciplina de Laboratório de Programação II é utilizar as principais estruturas de dados para solucionar problemas ligados a Computação.

- Presença e chamada:
  - Obrigatória podendo faltar até **7 aulas** (75% de 60h).
- Programa da disciplina (oficial)
- Calendário Acadêmico UFSM

2 Trabalhos

2.1 T1 - Pontos e distâncias

O objetivo deste T1 é **ler dois pontos em cada linha do arquivo de entrada e imprimir a distância**. Cada par de pontos é uma coordenada 2D `(x, y)`. O programa fará a leitura de um arquivo texto: `pontos.txt`. O arquivo terá o formato:

```
7.70 3.57 0.38 9.52
0.39 5.84 2.28 1.13
4.71 7.80 0.19 1.71
4.10 5.94 5.14 0.75
6.42 6.92 4.12 6.81
8.94 8.75 6.46 9.32
4.97 3.20 2.96 8.42
8.15 6.08 9.29 3.00
10.00 0.19 4.77 7.45
7.82 3.16 3.45 6.84
```

A saída do programa deve ser:

```
$ ./t1
Distancia 1 9.43
Distancia 2 5.07
Distancia 3 7.59
Distancia 4 5.30
Distancia 5 2.31
Distancia 6 2.55
Distancia 7 5.60
Distancia 8 3.28
Distancia 9 8.95
Distancia 10 5.71
```

Nas dicas abaixo, procure por estruturas de dados e entrada e saída.

ENTREGA: 19/08/2018, veja como entregar em Entrega de trabalhos.

Regras:

- Usar somente C++!
- Podem utilizar `std::vector`
- Devem fazer uso de alguma estrutura de dados (`struct`)
- Não compila, zero

- Plágio, zero

2.2 T2 - Editor com alocação dinâmica

Este trabalho consiste na implementação/alteração uma estrutura de dados do tipo **Editor** para suportar alocação dinâmica.

O programa exemplo será o código abaixo:

```
#include <iostream>
#include <vector>
#include <fstream>

// Estrutura  do Editor
struct Editor {
  std::vector<char*> linhas;

  void insere(std::string& linha)
  {
    // incluir alocação dinâmica aqui
    linhas.push_back( linha.c_str() );
  }

  char* remove(void)
  {
    char* l = linhas.back(); // pega ultima linha
    linhas.pop_back(); // remove ultima linha
    return l;
  }

  int tamanho(void)
  {
    return linhas.size();
  }

  void destroi(void)
  {
    // libera toda a memória
  }
}; // fim do editor

// programa
int main(int argc, char** argv)
{
  std::string linha;
```

```
  Editor editor;

  std::ifstream entrada{"texto.txt"};
  while(std::getline(entrada, linha)){
    editor.insere(linha);
  }

  auto tamanho = editor.tamanho();
  for(auto i=0 ; i < tamanho; i++){
    linha = editor.remove();
    std::cout << "DEL: " << linha << std::endl;
  }

  return 0;
}
```

Faça todas as alterações necessárias para suportar alocação dinâmica de memória (`new`) e liberação (`delete`). A correção utilizará principalmente o `valgrind`.

ENTREGA: 28/08/2019, veja como entregar em Entrega de trabalhos.

Regras:

- Usar somente C++!
- Obrigatório: `new` e `delete`
- Evite alocar e liberar memória dentro da `struct`
- Não compila, zero
- Plágio, zero

2.3 T3 - Editor gráfico Caca

Este trabalho consiste em terminar o **T3** com um editor de texto gráfico que utiliza alocação dinâmica de memória.

Os comandos básicos são:

- ESQ: sair do editor.
- CTRL+s: gravar arquivo e sair do modo edição.
- ENTER: quebra linha.

Outros comandos podem ser adicionados.

A estrutura do texto deve ser igual a anterior, ou seja, deve utilizar `new` e `delete`. As linhas do texto devem continuar dentro de um `std::vector`.

O laço principal do programa será:

```
int main(int argc, char **argv) {
  Editor editor;

  editor.inicia();
  editor.carrega("texto.txt");
  editor.legenda();

  while (editor.verifica_fim() == false) {
    editor.atualiza();
  }
  editor.salva("saida.txt");
  editor.finaliza();

  return 0;
}
```

As funções mínimas para o editor são:

- `carrega()` : carrega um arquivo texto.
- `salva()` : salva o conteúdo em um arquivo texto.
- `move_esq()`, `move_dir()`, `move_cima()`, `move_baixo()` : Devem ser completadas nos exemplos dados. Move a posição do cursor na direção indicada pelo nome da função. Para a esquerda, para cima e para baixo, não deve sair dos limites do texto. Para a direita, não há limite.
- `insere_char()` : Insere o caractere c na posição do cursor. Deve alocar uma nova região de memória grande o suficiente para conter a linha após a inserção, e copiar o conteúdo da linha antiga, alterada com a inserção do caractere novo. Caso a linha do cursor seja mais curta que a posição do cursor, devem ser inseridos espaços no final da linha, para que o caractere inserido fique na posição correta. A memória ocupada pela linha antiga deve ser liberada e o ponteiro da linha no vetor alterado para apontar para a nova região de memória com a nova versão da linha. Após a inserção, o cursor deve ser movido uma coluna à direita (usar a função acima).
- `remove_char()` : Remove o caractere na posição do cursor no texto apontado por txt. Como na inserção, deve alocar uma nova área de memória para conter a nova versão da linha. Caso o cursor esteja além do final da linha, a função não deve fazer nada. A posição do cursor não deve ser alterada.
- `gruda_linha()` : Se o cursor não estiver na última linha do texto, anexa a linha seguinte à do cursor ao final da linha do cursor, e remove a linha abaixo do cursor da lista.
- `quebra_linha()` : Quebra a linha do cursor na posição do cursor. A linha com o cursor terminará com o caractere logo antes do cursor, e a linha seguinte conterá os caracteres a partir da posição do cursor. Se o cursor estiver além do final da linha, a linha do cursor permanece inalterada e é inserida uma linha vazia após o cursor. Deve ser realocada a memória para a linha do cursor (se ela mudar) e liberada a memória antiga, alocada memória para a nova linha, inserida a nova linha no vetor de linhas. Editor não pode estar em modo edição.
- `recorta_linha()` : Recorta a linha onde está o cursor e armazena em uma área de transferência. A linha atual deve ser removida.
- `cola_linha()` : Se uma linha está na área de transferência, cola ela abaixo da posição do cursor.

Detalhes que serão considerados na avaliação:

- Todas as funções devem ser encapsuladas dentro da `struct`.
- A função `main()` do exemplo acima **deve ser igual** e não deve ser alterada.
- As variáveis que controlam a tela da `libcaca` devem estar dentro da `struct` Editor.
- Crie funções de tela dentro da `struct`, evite chamar funções da `libcaca` diretamente.

A implementação deve utilizar a **libcaca**. Baixe o programa exemplo aqui. É necessário instalar a biblioteca libcaca para gráficos. Em um sistema Ubuntu digite:

```
sudo apt install libcaca
```

Documentação em: http://caca.zoy.org/doxygen/

No Windows, procure no site http://caca.zoy.org/wiki/libcaca. Quem utiliza Codeblocks, adicione ao projeto todos os arquivos. Quem utilizar um sistema Linux pode compilar o programa digitando:

```
g++ -Wall -std=c++11 -o exemplo-caca exemplo-caca.cpp -lcaca
```

ENTREGA: 16/09/2019 com avaliação em aula. Veja como entregar em Entrega de trabalhos.

Regras:

- Usar somente C++!
- Obrigatório: `new` e `delete`.
- Use o valgrind.
- Não mande os binários.
- Não compila, zero.
- Plágio, zero.

2.4 Entrega de trabalhos

Crie um repositório **privado** e que apenas você tem acesso no site:

- https://github.com/

Adicione o usuário `@joao-lima` como colaborador do projeto.

Em cada trabalho, crie uma pasta nomeada `TX` sendo `X` o número do trabalho, por exemplo `T1`. Quando solicitado, envie **apenas** o link do repositório no Moodle da UFSM.

Cada trabalho será avaliado com base na avaliação de aula e nos códigos disponíveis no GitHub.

3 Aulas

A disciplina tem enfase em trabalhos práticos tendo poucas aulas teóricas. As duas primeiras aulas tem por objetivo apresentar a linguagem C++:

1. Introdução a C++
2. Memória

4 Material de apoio

4.1 Introdução C++

Nesse semestre, vamos utilizar a linguagem C++ na disciplina. Veja a página dedicada em Introdução em C++.

IMPORTANTE: A parte de orientação a objetos será ignorada.

4.2 Depuração de programas

- GDB: link 1, link 2, link 3. Ou a versão gráfica: DDD
- Valgrind: link 1, link 2.

4.3 Git

- GitHub: https://guides.github.com/introduction/git-handbook/
- Git Cheat Sheet: https://github.github.com/training-kit/
- Tech Talk: Linux Tolvards on git: http://youtu.be/4XpnKHJAok8
- Introduction to Git (local): GitIntro.pdf

5 Programa

Author: João Vicente Ferreira Lima
Created: 2019-08-28 qua 14:23
Validate