Projeto Prático - Sistema de Cadastro em Dispositivo Embarcado

1. Observações Iniciais

O objetivo deste trabalho é implementar um sistema de cadastro (com recursos para busca e alteração de dados) em dispositivos embarcados, preferencialmente Arduino, com uso de estruturas de dados diversas. O projeto vale 21 pontos e deverá ser realizado em equipe com dois a três alunos, sendo possível montar equipes com membros de turmas diferentes. A avaliação do projeto não será feita apenas sobre o código entregue mas também com a entrevista realizada com os membros de cada grupo. O projeto prevê uma única etapa, com apresentação parcial opcional em data especificada no campus virtual.

As seções a seguir detalham os requisitos e objetivos do projeto.

2. Instruções gerais para desenvolvimento e entrega do projeto prático:

Seguem as instruções e recomendações gerais para o desenvolvimento do projeto prático:

- Os arquivos fonte devem ser implementados em C++, usando técnicas de orientação a objetos (ou seja, os elementos principais devem ser implementados como classes).

- Não é permitido o uso da STL ou bibliotecas similares, sem a autorização dos professores. Isso é válido principalmente para elementos considerados centrais no trabalho.

- Nada de programação não-estruturada, não queremos ver um único "`goto`" no trabalho. E "`break`" só é aceito em "`switch`" (qualquer outro uso deve ser justificado antes com os professores). Também é recomendável não usar "`continue`". Variáveis globais que poderiam ser evitadas também reduzem grandemente o valor de seu trabalho.

- Portabilidade é questão chave: você deve programar lembrando-se que sua implementação será corrigida em linux. Portanto: nada de "`conio.h`" ou funções cheias de frescuras que não são portáveis (e que, geralmente, nada contribuem para o que realmente importa na resolução). Caso você precise realmente usar uma função que não é portável, verifique anteriormente com o professor o que pode ser feito neste caso. Não é exigido interface gráfica, mas se você quiser implementá-la, faça-o de forma que o sistema final fique portável.

- Boa organização, comentários bem feitos e indentação do arquivo contam pontos na nota final. Má organização e indentação retiram pontos, assim como comentários inúteis e sem sentido. Vocês tem liberdade para escolherem o estilo de indentação que julgarem mais adequado, mas respeitem o estilo escolhido. Recomendamos o uso de espaços para indentação, mas você pode usar tabulações, desde que não misture os dois tipos.

- O projeto deverá ser implementado preferencialmente com compilação separada, com classes principais em arquivos separados da aplicação. O uso de `Makefile` também é recomendado.

- -> Cada etapa deve ser entregue em arquivo compactado (preferência por zip, tar.gz ou tar.bz), sendo que o nome do arquivo deve conter os nomes dos membros da dupla e etapa do projeto, por exemplo: `joao_maria_etapa2.tar.gz`.

- Todo código-fonte deve ter um cabeçalho que permita identificar a equipe e a utilidade do arquivo, por exemplo:

```
/*
  Trabalho de Introdução à Estripotologia
    (nome da disciplina)
  Método de Zumbi-Xupacabra
     (aqui vai o nome do seu trabalho)
  Copyright 2016 by Sicrano Beltrano Fulano
     (aqui vai o seu nome)
  Arquivo que não faz nada
     (caso seu trabalho tenha vários arquivos,
      informe o que cada arquivo faz, qual a sua função)
*/
```

- Uma implementação que cumpre a obrigação recebe nota 80. Se você quer mais que 80, precisa avançar do que foi pedido, por exemplo:
  - fazendo uma boa interface, que seja prática (usando passagem de parâmetro no terminal por exemplo, caso a questão permita);
  - utilizando recursos de programação não solicitados, como controle de exceção, uso de templates, etc.
  - entrega antecipada;
  - implementação de recursos não solicitados, que mostrem esforço da equipe;
  - relatório bem escrito;
  - uso de compilação separada e makefiles;
  - etc.

Em cada etapa, além do código fonte, **deverá ser entregue um relatório e um arquivo de dados de exemplo**. O relatório deverá obrigatoriamente conter:

- Uma descrição de todas as estruturas de dados utilizadas;
- Uma descrição **em alto nível** (isto é, sem código) explicando a lógica do programa.

O arquivo de dados de exemplo deverá conter ao menos 40 itens cadastrados.

O trabalho deverá ser implementado utilizando C++ e ser passível de compilação e execução em um Linux genérico qualquer.

3. Objetivo do Projeto e Temas Disponíveis

Sua equipe deverá implementar um sistema de cadastro (com recursos para busca e alteração de dados) em dispositivos embarcados, preferencialmente Arduino, com uso de estruturas de dados diversas. Para isso, você deverá utilizar o Arduino com um cartão de memória.

Ao ser aberto, o programa abrirá o arquivo e permitirá as seguintes operações:

1. Inserir um novo objeto no arquivo.

2. Remover um objeto do arquivo. Fica a critério do grupo não apagar o objeto diretamente, mas marcar o espaço para reutilização. Nesse caso, a inserção deverá obrigatoriamente reutilizar espaços disponibilizados por remoção.

3. Consultar a um objeto no arquivo, usando busca binária ou sequencial.

4. Imprimir o arquivo, com todo seu conteúdo, na ordem de armazenamento, por meio da porta serial.

5. Imprimir os registros de modo ordenado, caso o armazenamento não seja feito de forma ordenada.

6. Possibilitar a ordenação dos registros por uma segunda chave.

Esclarecemos que essas ações não implicam obrigatoriamente em interface. As ações devem estar previstas, de alguma maneira, na aplicação, e devem, inclusive, fazer sentido para a aplicação planejada. Ou seja, não é necessário qualquer entrada e saída de dados padrão neste projeto.

Mais importante que essas ações é o fato que a aplicação deverá obrigatoriamente utilizar ao menos duas estruturas de dados distintas, preferencialmente as apresentadas na disciplina (consulte os professores para uso de outras estruturas). O uso das estruturas deverá ser argumentado no relatório da aplicação. Caso uma das estruturas não seja focada em arquivos, então a aplicação deverá fazer uso adicional de método de ordenação em disco. É importante que os dados armazenados possuam ao menos duas chaves (ex: data, nome, etc.), para permitir o ordenamento por uma chave secundária.

Não há limitações para temas, desde que os temas, propósito e estruturas não se repitam. Nossa sugestão é utilizar dados de sensores do próprio Arduino. Tão logo

sua equipe defina um tema e uma estrutura, informe em local próprio definido para esse propósito no Campus Virtual.

4. Projeto Alternativo

Tendo em mente que parte dos alunos não possuem afinidade com dispositivos embarcados e podem estar envolvidos em projetos que podem fazer melhor uso do conhecimento apresentado na disciplina, os professores irão, extraordinariamente, aceitar projetos que não envolvam obrigatoriamente o Arduino, mas que utilizem efetivamente e com boa justificativa estruturas de dados apresentadas. Isso também é motivado pelo pioneirismo do tema.

Nesse caso, o grupo deverá enviar a proposta aos professores, para que a mesma seja avaliada. Note: é importante que o sistema desenvolvimento seja realmente interessante ou relacionado ao tema de trabalho do aluno (como parte de estágio, IC, etc.) e que as estruturas realmente agreguem algo à aplicação.

5. Datas Importantes

Prazo final para definição da equipe e do tema*: 5 de outubro de 2018
* O tema inclui não apenas o assunto, mas pelo menos uma das estruturas a serem utilizadas no projeto.

Apresentação parcial opcional: (agendada com os professores) de 5 a 15 de novembro de 2018.

Entrega do trabalho: 5 de dezembro de 2018. Apresentações serão agendadas a partir desta data.