Estrutura de Dados I – Roteiro de Exercícios

prof. Andres Jessé Porfirio

Este roteiro de exercícios engloba os seguintes conceitos: Structs, Ponteiros, Pilha e Biblioteca. Utilizar os arquivos disponíveis no anexo do roteiro de exercícios, eles fornecem uma base para a implementação e desenvolvimento dos exercícios. Note que o arquivo stack.c está vazio, adicione nele a sua implementação de Pilha com alocação dinâmica (de acordo com o arquivo stack.h fornecido nos anexos).

1) Sabe-se que os alunos das escolas municipais de Guarapuava frequentemente participam de excursões para diversos lugares. Você foi encarregado de construir um programa que simule algumas condições destas excursões, para isto, deverá demonstrar conhecimentos em diversos tópicos abordados até o momento na disciplina de Estrutura de Dados I. Inicialmente, descompacte e analise os arquivos em anexo, tenha certeza de ter entendido a funcionalidade de cada um deles e, além disso, realize a compilação do projeto (preferencialmente via Makefile). O resultado esperado, após a execução do projeto compilado, será uma simples frase no console: “Seu código aqui!”.
2) A primeira alteração no projeto será a criação de structs e definição de tipos para realizar o controle dos alunos que embarcam no ônibus. Localize o arquivo stack.h e crie uma struct “aluno” cujo tipo de dado é “Aluno”. Todo aluno deve conter um nome (cadeia de caracteres) e uma matrícula (int).
3) Sabe-se que as excursões são oferecidas a todos as escolas da cidade, então crie também uma struct “escola”, cujo tipo de dado é “Escola” para representar as instituições. Toda escola deve conter um nome (cadeia de caracteres), um endereço (cadeia de caracteres) e um telefone (int).
4) Faça com que todo aluno possua uma escola associada, ou seja, inclua, na struct “aluno”, um ponteiro para uma variável do tipo “Escola”.
5) No arquivo “resolucao_exercicio.c”, inclua funções para imprimir “Aluno” e “Escola”. A função que imprime um aluno deve receber como parâmetro um ponteiro para uma variável do tipo “Aluno”, fazer sua impressão no console e não retornar dado algum. Faça o mesmo para o tipo de dado “Escola”.
6) Como todo “Aluno” possui uma variável “Escola*”, faça com que toda vez que um aluno for impresso, também seja impressa a “Escola” a qual ele pertence.
7) Na função principal do arquivo “resolucao_exercicio.c” inclua a inicialização de duas Escolas e 10 Alunos para testar as funções criadas no exercício 5. Faça a inicialização dos dados no próprio arquivo (não é necessário usar scanf..), utilize algo similar ao seguinte trecho de código:

```
Escola* escola1 = (Escola*)malloc(sizeof(Escola));
escola1->nome = “Escola Municipal 1”;
escola1->endereco = “Rua X, nro Y, BairroZ.”;
escola1->telefone = 99998888;

Aluno* a1 = (Aluno*)malloc(sizeof(Aluno));
a1->nome = “Joãozinho”;
a1->escola = escola1;
```

8) Realize a impressão de todos os alunos (note que as escolas devem ser impressas automaticamente.
9) O próximo passo é simular a entrada e saída dos alunos no ônibus escolar, sabe-se que o motorista é muito criterioso, e exige que o primeiro aluno a entrar se posicione no último banco do ônibus e, consequentemente, o último aluno a entrar fique no primeiro banco, assim, mantém-se a ordem de entrada e saída dos alunos, o último que entra é o primeiro a sair, sem gerar tumulto. Você, aluno esperto, notou que a melhor saída para esta simulação é utilizar uma estrutura de Pilha para organizar os alunos. No arquivo “stack.h”, altere o tipo de dado da Pilha para que armazene ponteiros para “Aluno” (e não mais char como a implementação padrão).
10) Altere o código da função principal (arquivo “resolucao_exercicio.c”) de modo que faça a simulação da viagem da excursão utilizando a pilha. Para cada aluno que entrar no ônibus (Pilha), imprima a seguinte mensagem no console: “Aluno: {nome_do_aluno} embarcou (push)”. Após acomodar todos os alunos no ônibus, imprima uma mensagem “Realizando a viagem...”. Em seguida, remova todos os alunos (pop). Para cada aluno que sai do ônibus imprima todos os seus dados (função criada no exercício 5). A saída do programa deve ter o seguinte formato:

```
Aluno: Fulano embarcou (push).
Aluno: Ciclano embarcou (push).
Aluno: Beltrano embarcou (push).

Realizando a viagem...

Aluno: Beltrano, Matricula: 3
       Escola Municipal 1, Rua X, nro Y, BairroZ. Fone: 99998888
Aluno: Ciclano, Matricula: 2
       Escola Municipal 1, Rua X, nro Y, BairroZ. Fone: 99998888
Aluno: Fulano, Matricula: 1
       Escola Municipal 2, Rua X, nro Y, BairroW. Fone: 77776666
```

11) No arquivo “stack.c”, analise a função void free_stack(Stack *S), note que ela possui um loop antes da liberação da memória relativa à Pilha, qual a finalidade destas instruções? O que aconteceria se esse loop não existisse? Crie um comentário na função respondendo estas questões, ex:

```
/*
* Sua explicação aqui.
*/
void free_stack(Stack *S)
{
    //...
}
```