Estruturas de Dados I - 2015.1
Data: 27/04/2015
Profa.: Adriana C. F. Alvim

Segundo Trabalho

Uma aplicação interessante de fila é a *ordenação por distribuição*, descrita a seguir. Seja uma lista L composta de n chaves, cada qual representada por um número inteiro em uma base $b > 1$. O problema consiste em ordenar essa lista. O algoritmo utiliza b filas, denotadas por $f_i, 0 \leq i \leq b-1$. Seja d o comprimento máximo da representação das chaves na base b. O algoritmo efetua d iterações, em cada uma das quais a lista L é percorrida. A primeira iteração considera o dígito da posição 1 (unidade) da chave. Se este for igual a k, a chave correspondente será removida da lista L e inserida na fila f_k. Ao terminar o percurso da lista L, esta se encontra vazia e distribuída pelas filas. Em seguida, deve-se incluir na lista L os elementos das filas em sequencia, isto é, primeiro os elementos de f_0, depois os de f_1, f_2, etc. Para a nova lista L, em ordem diferente da original, o processo deve ser repetido levando-se em consideração o segundo dígito da chave, e assim sucessivamente até que tenham sido feitas tantas distribuições quantos são os dígitos na chave de ordenação. Note que é necessário conhecer d, o número máximo de dígitos da lista L.

A seguir, um exemplo onde $b = 10$ e $d = 2$:

Seja `L = (19, 13, 5, 27, 1, 26, 31, 16, 2, 9, 11, 21, 60, 7)`.

Na primeira iteração é feita a primeira distribuição, das unidades, e as k filas ficam assim:

f_0 : 60
f_1 : $1, 31, 11, 21$
f_2 : 2
f_3 : 13
f_4 :
f_5 : 5
f_6 : $26, 16$
f_7 : $27, 7$
f_8 :
f_9 : $19, 9$

Após a concatenação das k filas a lista L fica assim:
`L = (60, 1, 31, 11, 21, 2, 13, 5, 26, 16, 7, 27, 19, 9).`

Em seguida, é feita a segunda iteração da segunda distribuição, das dezenas, e as k filas ficam assim:

f_0 : $1, 2, 5, 7, 9$
f_1 : $11, 13, 16, 19$
f_2 : $21, 26, 27$
f_3 : 31
f_4 :
f_5 :
f_6 : 60
f_7 :
f_8 :
f_9 :

Após a concatenação das k filas a lista L fica assim:
`L=(1, 2, 5, 7, 9, 11, 13, 16, 19, 21, 26, 27, 31, 60).`

Neste exemplo foram necessárias duas iterações pois o maior número da lista L tinha dois dígitos.

Desta forma, pede-se para implementar uma aplicação de Ordenação por Distribuição. Para tal, deve-se escolher uma estrutura de dados. Considere a classe `lista` conforme definda a seguir:

```
class lista{

protected:

  struct elo{
    int dado;
    elo * prox;
    elo():prox(NULL){};
    elo(int elem, elo *prox_elem=NULL):
    dado(elem),prox(prox_elem) {}
  };

  elo *prim;  /* ponteiro pra primeiro elemento */
  /* Calcula a qtd de digitos de um inteiro */
  int qtd_digitos(int elem);

public:
  lista():prim(NULL) {};
  ~lista();
  bool vazia();
  void insere(const int& novo);
  bool remove(int& elem);
  int maior_comprimento();
  void imprime() const;
};
```

A seguir, a funcionalidade das funções da classe `lista`:

a. Função `vazia`, que determina se a lista está vazia ou não.

b. Função `void insere(const int& novo)`, que insere `novo` no final da lista.

c. Função `bool remove(int& elem)`, que remove o primeiro elemento da lista e retorna seu valor na variável `elem`. Retorna `true` se a operação foi realizada com sucesso e `false` caso contrário.

d. Função `int maior_comprimento()`. Esta função percorre a lista de inteiros que chamou a função e, para cada elemento da lista, calcula o comprimento do elemento, armazenando o valor do maior comprimento entre todos os elementos. No final, a função retorna o maior comprimento de inteiro entre todos os elementos da lista. A função usa a função auxiliar `int qtd_digitos(int elem)` que retorna a quantidade de dígitos de um inteiro.

f. Função `imprime()`, que imprime os elementos da lista.

Em seguida, crie uma classe chamada `orddist` conforme mostrada a seguir:

```
class orddist{

protected:

  int b;        /* base */
  int d;        /* comprimento máximo do digito */
  lista* filas; /* vetor de listas */
  lista L;      /* lista inicial e que sera ordenada */

public:
  orddist(int base = 10);
  void carrega();
  void ordena();
};
```

A classe `orddist` deve possuir os seguintes atributos:

- b do tipo `int`: identifica a base, no nosso caso vamos fazer para uma base fixa, igual a 10.
- d do tipo `int`: identifica o comprimento máximo da representação das chaves na base b.
- L do tipo `lista`: armazena a lista que se deseja ordenar.
- $filas$ do tipo vetor de b listas do tipo `lista`: onde, por exemplo, o elemento `filas[0]` armazena a fila f_0, o elemento `filas[(b-1)]` armazena a fila f_{b-1}. Note que as filas são listas encadeadas com a particularidade que a função insere, insere no fim, e a função remove, remove do início.

Além do `constructor` e do `destructor`, implemente as seguintes funcionalidades para a classe `orddist`:

a. O construtor da classe atualiza o valor de $b = 10$ e inicializa o vetor `filas` com b elementos do tipo `lista`.

b. Função `void carrega()`. Esta função lê, via teclado, uma lista de números e insere em L. Em seguida, calcula o maior comprimento da representação das chaves na base b e armazena seu valor no atributo d da classe. Para tal, chama a função `maior_comprimento` da classe `lista`.

c. Função `void ordena()`. Esta função faz a ordenação propriamente dita.

Exemplo de programa `main`:

```
#include "lista.h"
#include "orddist.h"
#include <cstdlib>
int main(){

  orddist o;

  o.carrega();
  o.ordena();

  system("pause");
  return 0;
}
```

Observações:

- Data de entrega: 18/05/2015 até as 22:00.
- O código fonte deve estar adequadamente comentado.
- O grupo deve usar os arquivos `lista.h` e `orddist.h` disponibilizadas no Moodle.
- Entregar pelo sistema Moodle **APENAS** os arquivos `lista.cpp`, `ordist.cpp`.
- Se seu trabalho não compilar a nota é zero.
- Teste bem seu programa!
- O trabalho pode ser feito em dupla.
- A nota do trabalho vale 30% da N2 e só conta se a nota da P2 $> 3,0$.

Referências

[1] Szwarcfiter, Jayme e Markenzon, Liliam. *Estruturas de Dados e seus Algoritmos*, LTC Editora, 1994.