Pilhas e filas

Além do Sedgewick (**sempre** leiam o Sedgewick), veja

- `http://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html`
- `http://www.ime.usp.br/~pf/algoritmos/aulas/fila.html`

Pilhas e filas

Definição 1. *Uma pilha é uma ADT que admite duas operações: empilhar (correspondente à inserção) e desempilhar, que remove o item mais recentemente empilhado.*

▷ Política *last-in-first-out*; empilhar = *push*, desempilhar = *pop*

Definição 2. *Uma fila é uma ADT que admite duas operações: enfileirar (correspondente à inserção) e retirar, que remove o item inserido na fila há mais tempo.*

▷ Política *first-in-first-out*; enfileirar = *put*, retirar = *get*

Filas: interface

```
/* QUEUE.h */
void QUEUEinit(int);
 int QUEUEempty();
void QUEUEput(Item);
Item QUEUEget();
```

Um cliente (para interface levemente expandida)

```
/* q_client.c */
#include <stdio.h>
#include <stdlib.h>
#include "Item.h"
#include "QUEUEb.h"
int main(int argc, char *argv[])
{
  int i, max = 0;
  int N = atoi(argv[1]);
  QUEUEinit(N);
```

Um cliente (para interface levemente expandida)

```
/* q_client.c */
[...]
  for (i=0; i<N; i++) {
    if(rand() >= RAND_MAX/2.0) {
      QUEUEput(i);
      if (QUEUEsize() > max) max = QUEUEsize();
    } else
      if (!QUEUEempty()) QUEUEget();
    QUEUEdump();
  }
  printf("Tamanho maximo da fila: %d\n", max);
  return 0;
}
```

Exemplo de execução

```
$ cat Item.h
/* Item.h */
#include <stdio.h>
#define ITEMshow(A) printf("%d ", A)
typedef int Item;
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ make q_client.o prog4.10b.o
gcc -g -I. -Wall -pedantic -ansi -c q_client.c
gcc -g -I. -Wall -pedantic -ansi -c prog4.10b.c
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ gcc q_client.o prog4.10b.o
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$
```

Exemplo de execução

```
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ a.exe 16

1
1 2
1 2 3
1 2 3 4
2 3 4
3 4
4
4 8
8
8 10
10

13

15
Tamanho maximo da fila: 4
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$
```

Filas: uma implementação (com listas ligadas)

```
/* prog4.10.c */
#include <stdlib.h>
#include "Item.h"
#include "QUEUE.h"

typedef struct QUEUEnode *link;
struct QUEUEnode { Item item; link next; };

static link head, tail;

link NEW(Item item, link next)
  { link x = malloc(sizeof *x);
    x->item = item; x->next = next;
    return x;
  }
```

Filas: uma implementação (com listas ligadas)

```
/* prog4.10.c */
[...]
void QUEUEinit(int maxN)
  { head = NULL; }

int QUEUEempty()
  { return head == NULL; }

void QUEUEput(Item item)
  {
    if (head == NULL)
      { head = (tail = NEW(item, head)); return; }
    tail->next = NEW(item, tail->next);
    tail = tail->next;
  }
```

Filas: uma implementação (com listas ligadas)

```
/* prog4.10.c */
[...]
Item QUEUEget()
  { Item item = head->item;
    link t = head->next;
    free(head); head = t;
    return item;
  }
```

Filas: uma implementação (com vetores)

```
/* prog4.11.c */
#include <stdlib.h>
#include "Item.h"

static Item *q;
static int N, head, tail;

void QUEUEinit(int maxN)
  { q = malloc((maxN+1)*sizeof(Item));
    N = maxN+1; head = N; tail = 0; }

int QUEUEempty()
  { return head % N == tail; }
```

Filas: uma implementação (com vetores)

```
/* prog4.11.c */
[...]
void QUEUEput(Item item)
  { q[tail++] = item; tail = tail % N; }

Item QUEUEget()
  { head = head % N; return q[head++]; }
```

Exercício

Exercício 3 (Simulação de uma fila)**.** *Considere* `max`*, o tamanho máximo da fila no programa* `q_client.c` *acima. Verifique experimentalmente que* `max` *parece ser basicamente da ordem de* $\sqrt{\mathrm{N}}$ *para valores grandes de* N*. Para tanto, modifique* `q_client.c` *para que ele imprima (apenas) a razão* $\texttt{max}/\sqrt{\mathrm{N}}$ *e execute esta variante para valores grandes de* N*.*

Tipos de dados de primeira classe

Definição 4. *Um tipo de dados é de primeira classe se podemos ter vários objetos deste tipo, podemos declarar variáveis deste tipo, e podemos atribuir valores deste tipo a variáveis.*

- ▷ A implementação de fila acima não define um tipo de primeira classe.

Filas como uma ADT de primeira classe

```
/* QUEUE1st.h */
typedef struct queue *Q;
void QUEUEdump(Q);
   Q QUEUEinit(int);
 int QUEUEempty(Q);
void QUEUEput(Q, Item);
Item QUEUEget(Q);
```

Um cliente

```
/* prog4.19.c */
#include <stdio.h>
#include <stdlib.h>
#include "Item.h"
#include "QUEUE1st.h"

#define M 10

int main(int argc, char *argv[])
  { int i, j, N = atoi(argv[1]);
    Q queues[M];
    for (i = 0; i < M; i++)
      queues[i] = QUEUEinit(N);
```

Um cliente

```
/* prog4.19.c */
[...]
    for (j = 0; j < N; j++)
      QUEUEput(queues[rand() % M], j);
    for (i = 0; i < M; i++, printf("\n"))
      while (!QUEUEempty(queues[i]))
        printf("%2d ", QUEUEget(queues[i]));
    return 0;
  }
```

Uma implementação

```
/* prog4.20.c */
#include <stdlib.h>
#include "Item.h"
#include "QUEUE1st.h"

typedef struct QUEUEnode *link;
struct QUEUEnode { Item item; link next; };

struct queue { link head; link tail; };

link NEW(Item item, link next)
  { link x = malloc(sizeof *x);
    x->item = item; x->next = next;
    return x;
  }
```

Uma implementação

```
/* prog4.20.c */
[...]
Q QUEUEinit(int maxN)
  { Q q = malloc(sizeof *q);
    q->head = NULL; q->tail = NULL;
    return q;
  }

int QUEUEempty(Q q)
  { return q->head == NULL; }
```

Uma implementação

```
/* prog4.20.c */
[...]
void QUEUEput(Q q, Item item)
  {
    if (q->head == NULL)
      { q->tail = NEW(item, q->head);
        q->head = q->tail; return; }
    q->tail->next = NEW(item, q->tail->next);
    q->tail = q->tail->next;
  }
```

Uma implementação

```
/* prog4.20.c */
[...]
Item QUEUEget(Q q)
  { Item item = q->head->item;
    link t = q->head->next;
    free(q->head); q->head = t;
    return item;
  }
```

Execução

```
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ cat Item.h
/* Item.h */
typedef int Item;
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ make prog4.19.o prog4.20.o
gcc -g -I. -Wall -pedantic -ansi -c prog4.19.c
gcc -g -I. -Wall -pedantic -ansi -c prog4.20.c
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$
```

Execução

```
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ gcc prog4.19.o prog4.20.o
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$ a.out 100
 0  5 17 25 33 36 38 43 51 56 62 64 69 94 95
11 12 31 42 46 50 93
 3  7 20 23 28 39 66 67 89
 1  2 13 16 26 41 59 68 83 91 99
29 34 37 40 52 53 60 61 78 82 88
14 35 49 54 72 73 84 90 97
 8  9 18 44 45 58 65 79 81 86 98
21 22 32 48 57 75 77 85 96
 6 15 24 92
 4 10 19 27 30 47 55 63 70 71 74 76 80 87
yoshi@RANDOM ~/Main/www/2006ii/mac122a/exx/QUEUEs
$
```

Exercícios

Exercício 5. *Implemente o tipo fila* (`Q`) *como uma ADT de primeira classe usando vetores. (Isto é, reescreva* `prog4.20.c` *usando vetores em vez de listas ligadas.)*

Exercício 6. *Escreva os programas* `STACK1st.h` *e* `STACK1st.c` *que implementam pilha como um tipo de primeira classe. Escreva uma variante de* `prog4.19.c` *que usa pilhas em vez de filas.*

Exercício 7. *Leia sobre as Torres de Hanoi, por exemplo, em*

`http://mathworld.wolfram.com/TowersofHanoi.html`

Como você poderia usar pilhas para simular este jogo?