第五章　实验一

模拟缓冲池（Buffer Pool）

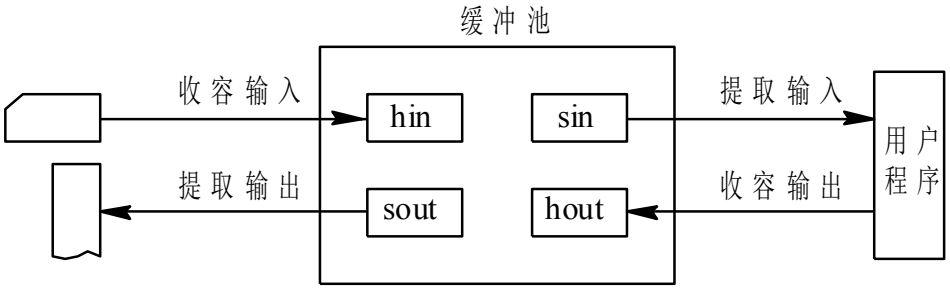
- **基本思想：** 对于既可用于输入又可用于输出的公用缓冲池， 其中至少应含有以下三种类型的缓冲区：**① 空(闲)缓冲区；②装满输入数据的缓冲区； ③装满输出数据的缓冲区。** 为了管理上的方便，可将相同类型的缓冲区链成一个队列，于是可形成以下三个队列：

  (1)空缓冲队列 emq：由空缓冲区所链成的队列。

  (2)输入队列 inq：由装满输入数据的缓冲区所链成的队列。

  (3)输出队列 outq：由装满输出数据的缓冲区所链成的队列。

- **四种工作缓冲区：**



- **⑴ 收容输入：**

  用于收容输入数据的工作缓冲区（hin）

  - 输入进程可调用 Getbuf(emq)过程，从空缓冲队列 emq 的队首摘下一空缓冲区，把它作为收容输入工作缓冲区 hin。
  - 然后，把数据输入其中，装满后再调用 Putbuf(inq，hin)过程，将它挂在输入队列 inq 队列上。

- **⑵ 提取输入：**

  用于提取输入数据的工作缓冲区（sin）

  - 计算进程可调用 Getbuf(inq)过程，从输入队列 inq 的队首取得一缓冲区，作为提取输入工作缓冲区 sin，计算进程从中提取数据。
  - 计算进程用完该数据后，再调用 Putbuf(emq，sin)过程，将它挂到空缓冲队列 emq 上。

- **⑶ 收容输出：**

  用于收容输出数据的工作缓冲区（hout）

  - 计算进程可调用 Getbuf(emq)，从空缓冲队列 emq 的队首取得一空缓冲，作为收容输出工作缓冲区 hout。
  - 当其中装满输出数据后，又调用 Putbuf(outq，hout)过程，将它挂在 outq 末尾。

- **⑷ 提取输出：**

  用于提取输出数据的工作缓冲区（sout）

  - 输出进程可调用 Getbuf(outq)过程，从输出队列的队首取得一装满输出数据的缓冲区，作为提取输出工作缓冲区 sout。
  - 在数据提取完后，再调用 Putbuf(emq，sout)过程，将它挂在空缓冲队列末尾。

实例调试：

- 三个队列 emq\inq\outq 的初始值

  初始状态：

```
空闲缓冲区队列emq内容:
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768


输入缓冲区队列inq:
Input buffer is empty.


输出缓冲区队列outq:
Output buffer is empty.
```

- 当 inq 为空队列时，“2 提取输入”不成功。

  输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
2
```

  输出：

```
Input your choice:
2

Input buffer is empty!Error!

空闲缓冲区队列emq内容:
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768


输入缓冲区队列inq:
Input buffer is empty.


输出缓冲区队列outq:
Output buffer is empty.
```

- 当 emq 不为空队列时，“1 收容输入—并输入 12”。

  输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
1

收容输入--请输入“输入数据”：12
```

输出：

```
空闲缓冲区队列emq内容：
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq：
 12

输出缓冲区队列outq：
Output buffer is empty.
```

- 当 emq 不为空队列时，“1 收容输入—并输入 24”。

输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
1

收容输入--请输入“输入数据”：24
```

输出：

```
空闲缓冲区队列emq内容：
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq：
 12  24

输出缓冲区队列outq：
Output buffer is empty.
```

- 当 inq 为非空队列时，“2 提取输入”。

输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
2
```

输出：

```
提取输入--输出“提取数据”：12
空闲缓冲区队列emq内容：
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq：
 24

输出缓冲区队列outq：
Output buffer is empty.
```

- 当 outq 为空队列时，“4 提出输出”不成功。

  输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
4
```

  输出：

```
Output buffer is empty!Error!

空闲缓冲区队列emq内容:
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq:
 24

输出缓冲区队列outq:
Output buffer is empty.
```

- 当 emq 不为空队列时，“3 收容输出—并输出数据-12”。

  输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
3

收容输出--请输入“输出数据”：-12
```

  输出：

```
空闲缓冲区队列emq内容:
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq:
 24

输出缓冲区队列outq:
 -12
```

- 当 emq 不为空队列时，“3 收容输出—并输出数据-24”。

  输入：

```
输出缓冲区队列outq:
 -12

what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
3

收容输出--请输入“输出数据”：-24
```

输出：

```
空闲缓冲区队列emq内容:
 -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq:
 24

输出缓冲区队列outq:
 -12  -24
```

- 当 outq 为非空队列时，“4 提取输出”。

输入：

```
what do you want to do?
1. 收容输入
2. 提出输入
3. 收容输出
4. 提出输出
5. 退出

Input your choice:
4
```

输出：

```
提取输出--输出“提取数据”：-12
空闲缓冲区队列emq内容:
 -32768  -32768  -32768  -32768  -32768  -32768  -32768  -32768

输入缓冲区队列inq:
 24

输出缓冲区队列outq:
 -24
```

- 使用 Micrsoft Visual Studio C++ 6.0 或 CodeBlocks 编程：程序 5_1_bufferpool.cpp。完善如下程序代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define Buff_MAX 10    //初始空闲缓冲区的个数
typedef struct buffernode
{
      int buf_type;
                  //"0"空闲缓冲区，"-1"装输入数据的缓冲区，"1"装输出数据的缓冲区
      int buf_data;                  //循环区数据
      struct buffernode *next;       //指向下一个指针
}node;
node *tempnode;   //全局变量

/*****************************************************************************
create 函数说明:建立空闲缓冲队列,每个空闲缓冲区内的数据存放“-32768”。
  返回值:     返回队列的头指针
*****************************************************************************/
node *creat( )     /*建立空闲缓冲队列*/
  { 填补程序 }

/*****************************************************************************
print_buf 函数说明:该函数打印三个缓冲队列的数据情况
  返回值:     无
*****************************************************************************/
void print_buf(node *emq, node *inq, node *outq)
   { 填补程序 }

/*****************************************************************************
Getbuf 函数说明: 该函数从队首读一个数据出来，使用 tempnode 指向队首第一个结点
  返回值:队列指针 head
*****************************************************************************/
node *Getbuf(node *head)
  { 填补程序 }

/*****************************************************************************
Putbuf 函数说明: 该函数从插入一个结点到队未。
  返回值:队列指针 head
*****************************************************************************/
node *Putbuf(node *head, node *newnode)
  { 填补程序 }
```

```
/**************MAIN 程序**********************/
void main(int argc, char *argv[])
{
    int flag;
    node *emq,*inq,*outq;
    node *hin,*sin,*sout,*hout;

    emq=creat();
    inq=NULL;
    outq=NULL;
    print_buf(emq,inq,outq);

    printf("\n#######################\n");
    printf("what do you want to do?\n");
    printf("1. 收容输入\n");
    printf("2. 提出输入\n");
    printf("3. 收容输出\n");
    printf("4. 提出输出\n");
    printf("5. 退出\n");
    printf("\nInput your choice:\n");
    scanf("%d",&flag);

    while(flag !=5)
    {
        switch(flag)
        { 填补程序 }
        printf("\n#######################\n");
        printf("what do you want to do?\n");
        printf("1. 收容输入\n");
        printf("2. 提出输入\n");
        printf("3. 收容输出\n");
        printf("4. 提出输出\n");
        printf("5. 退出\n");
        printf("\nInput your choice:\n");
        scanf("%d",&flag);
    }
}
```