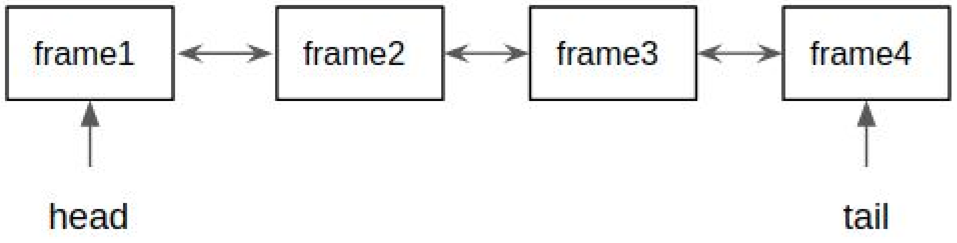
MP2 Design

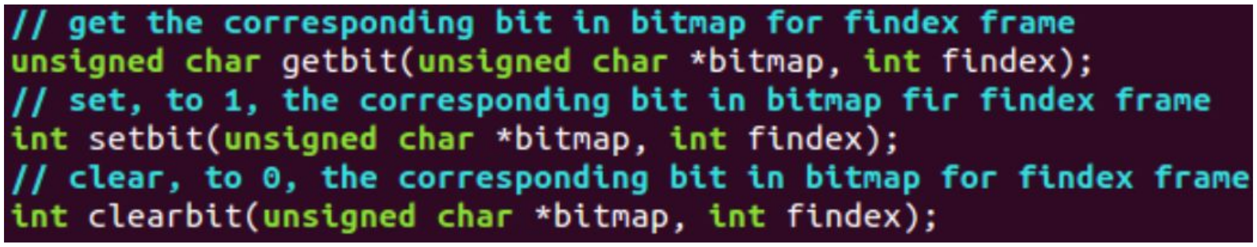
1. Class structure
(1)A static double linklist to manage all the frame pools.
I use static head and tail pointers to maintain this linklist, and each frame pool object also has prev and next pointers.



(2)2 bitmaps to manage the frames inside a frame pool
There are 3 states for each frame inside a frame pool: available, used as the head of the sequence, used not as the head
bitmap1 maintains whether the frame is used, 1 for available(def), 0 for used.
bitmap2 maintains whether the frame is used as a head or not, 1 for not as head(def), 0 for head

bitmap1	bitmap2	state
1	x	available for allocation
0	0	used as the head of a sequence frames
0	1	used, but not as the head

(3)bitmap operation utilities
getbit, get the bit value, 0/1, based on the bitmap and frame number as inputs
setbit, set the corresponding bit to 1, based on the bitmap and frame number as inputs
clearbit, clear the corresponding bit to 0, based on the bitmap and frame number as inputs

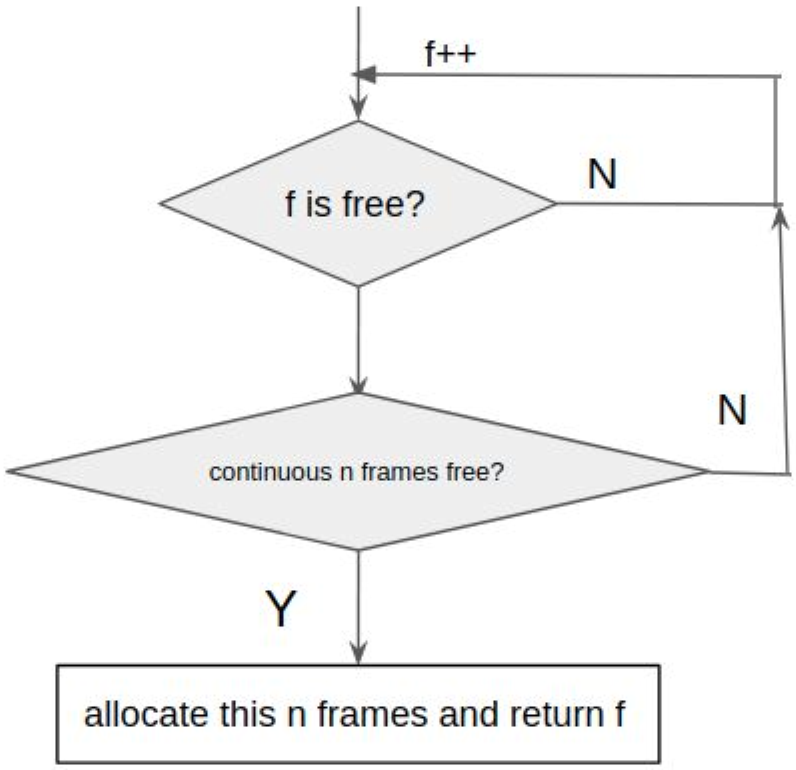


```
// get the corresponding bit in bitmap for findex frame
unsigned char getbit(unsigned char *bitmap, int findex);
// set, to 1, the corresponding bit in bitmap fir findex frame
int setbit(unsigned char *bitmap, int findex);
// clear, to 0, the corresponding bit in bitmap for findex frame
int clearbit(unsigned char *bitmap, int findex);
```

2. Core function implementation
(1)Constructor
ContFramePool::ContFramePool

Initialization, validation checking and set up the attributes, including base frame number, number of frames, bitmap allocation and set all the frames to be available for allocation, and if the bitmaps are inside this frame, also mark that bits to be used.
Also update the static double linklist.

(2)get_frames(_n_frames)
Traverse bitmap1 to look for a sequence of _n_frames available to allocate. If discovered, mark them as used in bitmap1 and also mark the first frame as head in bitmap2, then return the first frame allocated.


f++
f is free?
N
continuous n frames free?
N
Y
allocate this n frames and return f

(3)mark_inaccessible(_base_frame_no, _n_frames)
Mark a sequence of _n_frames to be used from free frames, beginning from _base_frame_no, and also mark _base_frame_no frame to be head.
If any of them is already used, it is an error.

(4)release_frames(_first_frame_no)
Check whether _first_frame_no is marked as used as head, if not, it is an error; if so, free it, and also free the following frames until the next frame is free or a head for another sequence.

(5)needed_info_frames(_n_frames)
Just compute the frames to store the bitmap1 and bitmap2 for _n_frames frames and return.
In my implementation, 1 info_frame can store the bitmap information for 8192 frames, so I compute it as: _n_frames/8192 + (_n_frames%8192>0 ? 1:0)