Main structures:

1)chunk

2)manager

Chunk:

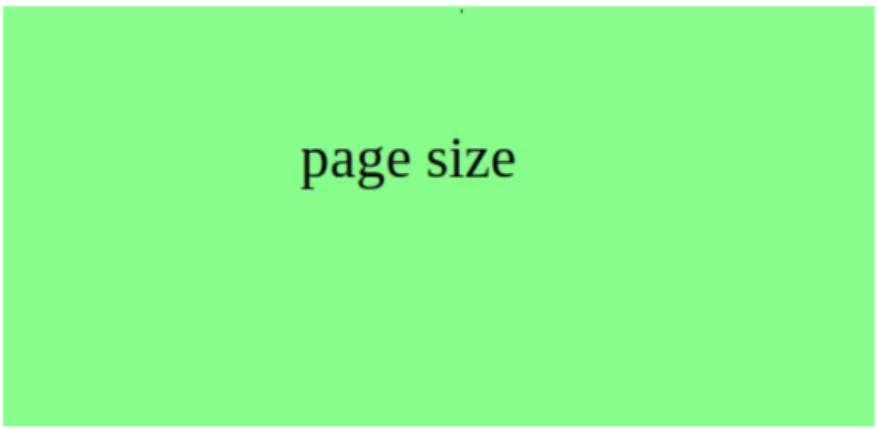
Chunk is a simple abstration above part of the memory. Chunks contain address of the block start, size and status (free or busy). The first two are divided by maximum alignment of current system (if object's size isn't divided by maximum alignment, system manager will give it more memory).

Manager:

Manager supervises memory and chunks. Manager is able to do two plain things: provide you with the new memory pointer for the objects and overwrite object's data with PRN.

Consider the "new" alorithm

Manager reserves random page and creates chunk when you put the first object in it. Chunk contains information about the whole page.



Next step is to find the proper chunk where manager can put the object. But how we know manager has only the one chunk now, so this current chunk be chosen. We will consider the case when there is more than one chunk later. Then manager counts offset from the beginning of the chunk and puts object in the middle of it. Offset depends on the object size and random number.

After this there are two possible ways. If offset doesn't equal to zero manager will remove greater chunk and create three smaller pieces of it. In other way, if offset equals zero or page size equals size of the object (sizeof(object)), manager will create 2 smaller chunks instead of three.

Now we are going to consider the case when there are already a few objects before insertion(!). Manager contains sub-object named "widness_of_choise" (by default it equals 10, but in subsequent versions it will be changeable). Sub-object shows amount of places required on the page to put object in it. We call places there size of free chunk divided by rounded size of current object. Time complexity of this algorithm is O(n) where "n" is the number of free chunks. It's achieved by "multi_index" that can provide access to the sorted chunks in two various methods: by state & size and by address.

When number of places is counted, there are two possible ways:

1) current number of places >= widness_of_choise

    In this case manager randomly finds chunks which are able to accommodate the object using considered algorithm

2) current number of places < widness_of_choise

    In this case there is a chance that manager will reserve one more page and use algorithm above again.

Consider the "delete" alorithm

This algorithm is pretty simple. Manager finds chunk with given address (with O(log(n)) time complexity). All the data provided by chunk becomes overwritten. After this, if left or right chunks next to be found by address on page exists and free, they become in one free chunk. If there are free chunk with size equals page size it will be deleted and page will be unmapped