一、 实验目的

使用 C++模拟分页存储管理

二、 数据结构及说明

```
#define max  100//物理块个数
#define page  1024//页面大小，单位 B
```

宏定义两个常量，分别表示物理块个数及每个页面大小

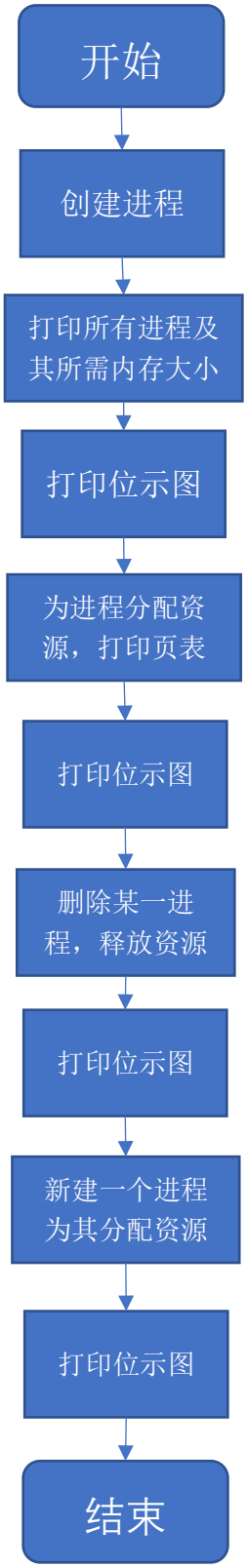
```
struct Page_Table {
        int page_num;//页号
        int block_num;//块号
        };
```

定义一个结构体，表示页表。

```
class PCB {
private:
        string name;
        int size;
        bool ifallo = false;
        vector<Page_Table> page_item;
 public:
        PCB(string n1, int s);
        void allo();//分配
        void show();//打印页表
        void del();//回收
        string get_name();//获取进程名
        int get_size();
        };
```

定义一个类，表示进程控制块 PCB，数据成员有进程名、进程所需大小。函数成员有，分配内存，打印页表，释放资源。

三、 流程图



四、 实验结果

进程创建完成，共有 12 个进程!
进程名 所需空间

H 553
G 2357
U 1725
E 3012
Y 540
N 1728
F 1348
X 1495
I 3507
C 850
S 1480
B 4414

位示图:

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

分配完成!
打印页表:
进程名：H 所需空间:553
页号 块号
0 0

进程名：G 所需空间:2357
页号 块号
0 1
1 2

2 3

进程名：U 所需空间:1725
页号 块号
0 4
1 5

进程名：E 所需空间:3012
页号 块号
0 6
1 7
2 8

进程名：Y 所需空间:540
页号 块号
0 9

进程名：N 所需空间:1728
页号 块号
0 10
1 11

进程名：F 所需空间:1348
页号 块号
0 12
1 13

进程名：X 所需空间:1495
页号 块号
0 14
1 15

进程名：I 所需空间:3507
页号 块号
0 16
1 17
2 18
3 19

进程名：C 所需空间:850
页号 块号
0 20

进程名：S 所需空间:1480

页号 块号
0 21
1 22

进程名：B 所需空间:4414
页号 块号
0 23
1 24
2 25
3 26
4 27

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

删除进程名为:Y 的进程，并释放所占用的物理块!
进程名：Y 所需空间:540
页号 块号
0 9

位示图:
0 1 1 1 2 1 3 1 4 1
5 1 6 1 7 1 8 1 9 0
10 1 11 1 12 1 13 1 14 1
15 1 16 1 17 1 18 1 19 1

20 1　　21 1　　22 1　　23 1　　24 1
25 1　　26 1　　27 1　　28 0　　29 0
30 0　　31 0　　32 0　　33 0　　34 0
35 0　　36 0　　37 0　　38 0　　39 0
40 0　　41 0　　42 0　　43 0　　44 0
45 0　　46 0　　47 0　　48 0　　49 0
50 0　　51 0　　52 0　　53 0　　54 0
55 0　　56 0　　57 0　　58 0　　59 0
60 0　　61 0　　62 0　　63 0　　64 0
65 0　　66 0　　67 0　　68 0　　69 0
70 0　　71 0　　72 0　　73 0　　74 0
75 0　　76 0　　77 0　　78 0　　79 0
80 0　　81 0　　82 0　　83 0　　84 0
85 0　　86 0　　87 0　　88 0　　89 0
90 0　　91 0　　92 0　　93 0　　94 0
95 0　　96 0　　97 0　　98 0　　99 0

新创建一个进程名为:K 所需空间大小为:1272 的进程!
为其分配空间:
进程名：K 所需空间:1272
页号 块号
　0　　9
　1　　28

位示图:
 0 1　　 1 1　　 2 1　　 3 1　　 4 1
 5 1　　 6 1　　 7 1　　 8 1　　 9 1
10 1　　11 1　　12 1　　13 1　　14 1
15 1　　16 1　　17 1　　18 1　　19 1
20 1　　21 1　　22 1　　23 1　　24 1
25 1　　26 1　　27 1　　28 1　　29 0
30 0　　31 0　　32 0　　33 0　　34 0
35 0　　36 0　　37 0　　38 0　　39 0
40 0　　41 0　　42 0　　43 0　　44 0
45 0　　46 0　　47 0　　48 0　　49 0
50 0　　51 0　　52 0　　53 0　　54 0
55 0　　56 0　　57 0　　58 0　　59 0
60 0　　61 0　　62 0　　63 0　　64 0
65 0　　66 0　　67 0　　68 0　　69 0
70 0　　71 0　　72 0　　73 0　　74 0
75 0　　76 0　　77 0　　78 0　　79 0
80 0　　81 0　　82 0　　83 0　　84 0
85 0　　86 0　　87 0　　88 0　　89 0
90 0　　91 0　　92 0　　93 0　　94 0

95 0　　96 0　　97 0　　98 0　　99 0

五、 结果分析

还未对进程分配空间时，位示图显示所有物理块都没用，为进程分配资源后，由于，是对物理块按顺序分配，所以，被使用的物理块是连续的，每个进程所分配的物理块也是连续的，从位示图可以看出，使用了 27 个物理块。删除进程名为 Y 的进程，释放其所占物理块（第 9 个物理块），新建一个进程 K，由于其所需大小为 1272B，所以需要占用两个物理块，由于第 9 个物理块是空的，所以先占用第 9 个物理块，然后再占用第 28 个物理块。