a.c如下：

#define screen ((char far \*)0xb8000000)

typedef struct c{

char chr;

char color;

void (\*put)(struct c \*,int,int);

}ch;

void f(ch \*,int,int);

main()

{

int n;

ch a;

a.chr='c';

a.color=2;

a.put=f;

a.put(&a,10,40);

}

/\* 函数f,在屏幕的row行和col列打印字符，row,col从1开始编号 \*/

void f(ch \*p,int row,int col)

{

screen[(row-1)\*160+(col-1)\*2]=p->chr;

screen[(row-1)\*160+(col-1)\*2+1]=p->color;

}

b.c如下：

#define screen ((char far \*)0xb8000000)

typedef struct c{

char chr;

char color;

void (\*setch)(struct c\*,char);

void (\*setcolor)(struct c\*,char);

void (\*put)(struct c \*,int,int);

}ch;

void f(ch \*,int,int);

void f1(ch \*,char);

void f2(ch \*,char);

main()

{

int n;

ch a;

a.put=f;

a.setch=f1;

a.setcolor=f2;

a.setch(&a,'c');

a.setcolor(&a,2);

a.put(&a,10,40);

}

/\* 函数f,在屏幕的row行和col列打印字符，row,col从1开始编号 \*/

void f(ch \*p,int row,int col)

{

screen[(row-1)\*160+(col-1)\*2]=p->chr;

screen[(row-1)\*160+(col-1)\*2+1]=p->color;

}

void f1(ch \*p,char a)

{

p->chr=a;

}

void f2(ch \*p,char color)

{

p->color=color;

}

从面向对象程序设计的角度来看，感觉a.c b.c对ch的封装都好。

补全new程序：

#define screen ((char far \*)0xb8000000)

#define new(x) ((x \*)(malloc(sizeof(x))))

typedef struct c{

char chr;

char color;

void (\*put)(struct c \*,int,int);

}ch;

void f(ch \*,int,int);

main()

{

int n;

ch \*a=new(ch);

a->chr='c';

a->color=2;

a->put=f;

a->put(a,10,40);

}

/\* 函数f,在屏幕的row行和col列打印字符，row,col从1开始编号 \*/

void f(ch \*p,int row,int col)

{

screen[(row-1)\*160+(col-1)\*2]=p->chr;

screen[(row-1)\*160+(col-1)\*2+1]=p->color;

}