Chessman.h

```
#ifndef CHESSMAN
#define CHESSMAN

#include <iostream>

class Figure
{
  protected:
    char pos[4];
  public:
    Figure(char*);
    Figure() {};
    virtual int attack(char*)=0;
    virtual char isa()=0;
    operator char*();
    static int deskout(char*);
    int danger();
    Figure& dodge(Figure*);
    int city(char*);
    Figure& operator=(char*);
    int operator==(char*);
    int operator!=(char*);
    friend std::ostream& operator<<(std::ostream&, Figure&);
    friend std::istream& operator>>(std::istream&, Figure&);
};
#endif
```

Chessman.cc

```
#include "chessman.h"
using std::cout;

Figure::Figure(char* p)
{
  pos[0] = p[0]; pos[1] = p[1]; pos[2] = pos[3] = '\0';
}

Figure::operator char*()
{
  return pos;
}

int Figure::deskout(char* p)
{
  return((p[0] > 'h') || (p[0] < 'a') || (p[1] < '1') || (p[1] > '8'));
}

Figure& Figure::operator=(char* p)
{
  pos[0] = p[0]; pos[1] = p[1];
  return(*this);
}

int Figure::operator==(char* p)
{
  return((pos[0] == p[0]) && (pos[1] == p[1]));
}

int Figure::operator!=(char* p)
{
  return((pos[0] != p[0]) || (pos[1] != p[1]));
}

std::ostream& operator<<(std::ostream& out, Figure& f)
{
  return out << f.isa() << f.pos[0] << f.pos[1];
}

std::istream& operator>>(std::istream& in, Figure& f)
{
  char s[2];
  std::cin.unsetf(std::ios::skipws);
  in >> s[0] >> s[1];
  in.ignore(64, '\n');
  if((f.attack(s) == 0) || (f == s))    // if(f.deskout(s))
    in.clear(std::ios::failbit | in.rdstate());
  f = s;
  return in;
}

int Figure::danger()
{
  int i, j;
  char s[2];
  char* mark = ".+x";
  char m;
  cout << "  a b c d e f g h\n";
  for(i=8; i > 0; i--)
  {
    cout << i << ' ';
    s[1] = '0' + i;
    for(j=0; j < 8; j++)
    {
      s[0] = 'a' + j;
      m = (*this != s) ? mark[attack(s)] : isa(); // attack ?
      cout << m << ' ';
    }
    cout << i << "\n";
  }
  cout << "  a b c d e f g h\n";
  return(0);
}

int Figure::city(char* p)
{
  int dx, dy;
  if((dx = pos[0] - p[0]) < 0)
    dx = -dx;
  if((dy = pos[1] - p[1]) < 0)
    dy = -dy;
  return(dx + dy);
}

Figure& Figure::dodge(Figure* f)
{
  char s[2];
  char p[2];
  char q[2];
  short d;
  short dm = 16;
  short i, j;
  p[0] = p[1] = q[0] = q[1] = 0;
  for(i=8; i > 0; i--)
  {
    s[1] = '0' + i;
    for(j=0; j < 8; j++)
    {
      s[0] = 'a' + j;
      if(*this == s)
        continue;
      if(!attack(s))
        continue;
      q[0] = s[0]; q[1] = s[1];
      if(f->attack(s))
        continue;
      if((d = f->city(s)) > dm)
        continue;
      dm = d; p[0] = s[0]; p[1] = s[1];
    }
  }
  *this = p[0] ? p : q;
  return(*this);
}
```

Chesslib.h

```
#ifndef CHESSLIB
#define CHESSLIB

#include "chessman.h"

class Plus : virtual public Figure
{
  public:
    Plus(char* p) : Figure(p) {};
    Plus() {};
    char isa() { return 'P'; };
    int attack(char*);
};

class Cros : virtual public Figure
{
  public:
    Cros(char* p) : Figure(p) {};
    Cros() {};
    char isa() { return 'C'; };
    int attack(char*);
};

class Star : public Plus, public Cros
{
  public:
    Star(char* p) : Figure(p) {};
    Star() {};
    char isa() { return 'S'; };
    int attack(char*);
};
#endif
```

Chesslib.cc

```
#include "chesslib.h"

int Plus::attack(char* p)
{
  if(deskout(p) > 0)
    return(0);
  int x = p[0] - pos[0];
  int y = p[1] - pos[1];
  if(x < 0) x = -x;
  if(y < 0) y = -y;
  if((x + y) < 2)
    return(1);
  return(0);
}

int Cros::attack(char* p)
{
  if(deskout(p) > 0)
    return(0);
  int x = p[0] - pos[0];
  int y = p[1] - pos[1];
  if(x < 0) x = -x;
  if(y < 0) y = -y;
  if(((x + y) > 2) || ( x != y))
    return(0);
  return(1);
}

int Star::attack(char* s)
{
  if(Plus::attack(s) > 0)
    return(1);
  if(Cros::attack(s) > 0)
    return(2);
  return(0);
}
```

chess+.cc

```
#include <iostream>
#include "chesslib.h"

int main(int argc, char* argv[])
{
  if(argc < 2)
    argv[1] = "nil";
  if(Figure::deskout(argv[1]))
  {
    std::cout << "Correct " << argv[1];
    std::cout << " to " << (argv[1] = "a1") << " Default\n";
    std::cout << "Usage example: chess+ e5\n";
  }
  Star f(argv[1]);
  do
  {
    f.danger();
    std::cout << f << '-' << f.isa();
  } while (std::cin >> f);
  return(0);
}
```