```
#include <stdio.h>

class nextIntFuncBase {
 public:

 // Dereference function pointer or call built-in function.
 virtual int operator ()(int lastInt) = 0;

 // Allow for old-style function pointer dereferencing.
 nextIntFuncBase & operator *() {
   return (*this);
 }
};

// C bound function pointer class.
class nextIntFuncCBFP : public nextIntFuncBase {
 typedef int (* nextIntFunc)(int lastInt);

 nextIntFunc _nextInt;

 public:

 nextIntFuncCBFP(nextIntFunc nextInt) : _nextInt(nextInt) {
 }

 // Dereference function pointer.
 virtual int operator ()(int lastInt) {
   return (_nextInt(lastInt));
 }
};

// C++ bound function pointer class.
class nextIntFuncCPPBFP : public nextIntFuncBase {
 FILE * _f;

 public:

 nextIntFuncCPPBFP(const char * fname) {
   _f = fopen (fname, "r");
 }

 ~nextIntFuncCPPBFP() {
   fclose (_f);
 }

 // Function is built in to class.
 virtual int operator ()(int lastInt) {
   int nextInt;
```

```
   if (fscanf(_f, "%d", &nextInt) == 1)
     nextInt += lastInt;
   else
     nextInt = 0;

   return (nextInt);
  }
};

// *** Note change in typedef from BFP1.C.
typedef nextIntFuncBase & nextIntFunc;

void printInt(nextIntFunc nextInt) {
 int value = 0;

 while ((value = (*nextInt)(value)) != 0)
 /*
  * Could also use newer style:
  * while ((value = nextInt(value)) != 0)
  */
   printf ("%d\n", value);
}

FILE * intFile;

int readFile(int lastInt) {
 int nextInt;

 if (fscanf(intFile, "%d", &nextInt) == 1)
   nextInt += lastInt;
 else
   nextInt = 0;

 return (nextInt);
}

int main() {
 puts ("Testing C-style BFP ...");
 intFile = fopen ("INT.DAT", "r");
 // *** Note change in call to printInt() from BFP1.C.
 printInt (nextIntFuncCBFP(readFile));
 fclose (intFile);

 puts ("Testing C++-style BFP ...");
 printInt (nextIntFuncCPPBFP ("INT.DAT"));

 return (0);
}
```