Standard I/O Operators

The prototypes for the overloaded insertion and extraction operators for standard input and output on objects of our own classes take the form

std::istream& operator>>(std::istream&, *Type*&);

std::ostream& operator<<(std::ostream&, const *Type*&);

where *Type* is the name of the class.

The header file for our Student class that includes their declarations is:

|  |
| --- |
| // Student.h  #include <iostream> // for std::ostream, std::istream  const int NG = 20;  class Student {  int no;  float grade[NG];  int ng;  public:  Student();  Student(int);  Student(int, const float\*, int);  void read(std::istream&);  void display(std::ostream& os) const;  };  std::istream& operator>>(std::istream& is, Student& s);  std::ostream& operator<<(std::ostream& os, const Student& s); |

The implementation file for our upgraded Student class contains:

|  |
| --- |
| // Student.cpp  #include "Student.h"  using namespace std;  Student::Student() {  no = 0;  ng = 0;  }  Student::Student(int n) {  \*this = Student(n, nullptr, 0);  }  Student::Student(int sn, const float\* g, int ng\_) {  bool valid = sn > 0 && g != nullptr && ng\_ >= 0;  if (valid)  for (int i = 0; i < ng\_ && valid; i++)  valid = g[i] >= 0.0f && g[i] <= 100.0f;  if (valid) {  // accept the client's data  no = sn;  ng = ng\_ < NG ? ng\_ : NG;  for (int i = 0; i < ng; i++)  grade[i] = g[i];  } else {  \*this = Student();  }  }  void Student::read(istream& is) {  int no; // will hold the student number  int ng; // will hold the number of grades  float grade[NG]; // will hold the grades  cout << "Student Number : ";  is >> no;  cout << "Number of Grades : ";  is >> ng;  if (ng > NG) ng = NG;  for (int i = 0; i < ng; i++) {  cout << "Grade " << i + 1 << " : ";  is >> grade[i];  }  // construct a temporary Student  Student temp(no, grade, ng);  // if data is valid, the temporary object into the current object  if (temp.no != 0)  \*this = temp;  }  void Student::display(ostream& os) const {  if (no > 0) {  os << no << ":\n";  os.setf(ios::fixed);  os.precision(2);  for (int i = 0; i < ng; i++) {  os.width(6);  os << grade[i] << endl;  }  os.unsetf(ios::fixed);  os.precision(6);  } else {  os << "no data available" << endl;  }  }  std::ostream& operator<<(ostream& os, const Student& s) {  s.display(os);  return os;  }  std::istream& operator>>(istream& is, Student& s) {  s.read(is);  return is;  } |

The following client code uses our upgraded Student class accepts the input shown on the right and produces the results shown below:

|  |  |
| --- | --- |
| // Standard I/O Operators  // standardIO.cpp  #include "Student.h"  int main () {  Student harry;  std::cin >> harry;  std::cout << harry;  } | Student Number : 1234  Number of Grades : 2  Grade 1 : 56.7  Grade 2 : 78.9  1234:  56.70  78.90 |