Лабораторная работа №7.

Текст программы для задания №6.

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48.  49.  50.  51.  52.  53.  54.  55.  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82. | #include <iostream>  #include <string>  #include <vector>  struct Abonent {  unsigned int \_number;  std::string \_secondName;  unsigned long int \_passNumber;  Abonent() {}  Abonent(unsigned int number, std::string secondName, unsigned long int passNumber) {  \_number = number;  \_secondName = secondName;  \_passNumber = passNumber;  }  void input() {  std::cout << "Input second name\n";  std::cin >> \_secondName;  std::cout << "Input number\n";  std::cin >> \_number;  std::cout << "Input number of passport\n";  std::cin >> \_passNumber;  }  void output() {  std::cout << "=======================================\n";  std::cout << "The abonent " << \_secondName << '\n';  std::cout << "Number: +" << \_number << '\n';  std::cout << "=======================================\n";  }  };  struct AbonentExtended : Abonent {  long double \_debtValue;  AbonentExtended() {}  AbonentExtended(unsigned int number, std::string secondName, unsigned long int passNumber, long double debtValue) :  Abonent(number, secondName, passNumber) {  \_debtValue = debtValue;  }  bool checkDebetor(long double base) {  return !(\_debtValue < base);  }  void input() {  std::cout << "Input second name\n";  std::cin >> \_secondName;  std::cout << "Input number\n";  std::cin >> \_number;  std::cout << "Input number of passport\n";  std::cin >> \_passNumber;  std::cout << "Input debt value\n";  std::cin >> \_debtValue;  }  void output() {  std::cout << "=======================================\n";  std::cout << "The abonent " << \_secondName << '\n';  std::cout << "Number: " << \_number << '\n';  std::cout << "Debt: " << \_debtValue << '\n';  std::cout << "=======================================\n";  }  };  int main() {  unsigned int N;  long double debtVal;  std::cin >> N;  std::vector<AbonentExtended> table = std::vector<AbonentExtended>(N);  for (auto it = table.begin(); it != table.end(); ++it) {  it->input();  }  system("cls");  std::cout << "Input debt base value: ";  std::cin >> debtVal;  std::cout << "Searching debetors...\n";  for (auto it = table.begin(); it != table.end(); ++it) {  if (it->checkDebetor(debtVal)) {  it->output();  }  }  return 0;  } |

**Оценка характеристик программы**

В тексте программы представлено 2 класса: Abonent и AbonentExtended. Рассчитаем вес каждого из классов:

Теперь рассчитаем количество операций, переопределяемых классом (NOO):

Далее рассчитаем значение NOA (количества операций, добавленных в класс):

Тогда Индекс специализации будет равен:

Теперь рассчитаем значение средннего размера операции AOS:

Далее рассмотрим сложность операций классов ОС на основе LOC-оценок:

Метод input класса Abonent:

|  |  |  |
| --- | --- | --- |
| Действие | Вес | Количество строк |
| Определение переменной-параметра | 0.3 | 3 |
| Определение временной переменной | 0.5 | 0 |
| Присваивание значения | 0.5 | 0 |
| Вложенное выражение | 0.5 | 0 |
| Сообщение без параметров | 1 | 0 |
| Арифм. операция | 2 | 0 |
| Сообщение с параметрами | 3 | 0 |
| Вызов стандартной функции интерфейса | 5 | 6 |
| Вызов пользовательской функции | 7 | 0 |

Метод output класса Abonent:

|  |  |  |
| --- | --- | --- |
| Действие | Вес | Количество строк |
| Определение переменной-параметра | 0.3 | 0 |
| Определение временной переменной | 0.5 | 0 |
| Присваивание значения | 0.5 | 0 |
| Вложенное выражение | 0.5 | 0 |
| Сообщение без параметров | 1 | 0 |
| Арифм. операция | 2 | 0 |
| Сообщение с параметрами | 3 | 0 |
| Вызов стандартной функции интерфейса | 5 | 4 |
| Вызов пользовательской функции | 7 | 0 |

Метод input класса AbonentExtended:

|  |  |  |
| --- | --- | --- |
| Действие | Вес | Количество строк |
| Определение переменной-параметра | 0.3 | 4 |
| Определение временной переменной | 0.5 | 0 |
| Присваивание значения | 0.5 | 0 |
| Вложенное выражение | 0.5 | 0 |
| Сообщение без параметров | 1 | 0 |
| Арифм. операция | 2 | 0 |
| Сообщение с параметрами | 3 | 0 |
| Вызов стандартной функции интерфейса | 5 | 8 |
| Вызов пользовательской функции | 7 | 0 |

Метод output класса Abonent:

|  |  |  |
| --- | --- | --- |
| Действие | Вес | Количество строк |
| Определение переменной-параметра | 0.3 | 0 |
| Определение временной переменной | 0.5 | 0 |
| Присваивание значения | 0.5 | 0 |
| Вложенное выражение | 0.5 | 0 |
| Сообщение без параметров | 1 | 0 |
| Арифм. операция | 2 | 0 |
| Сообщение с параметрами | 3 | 0 |
| Вызов стандартной функции интерфейса | 5 | 5 |
| Вызов пользовательской функции | 7 | 0 |

Метод checkDebetor класса AbonentExtended:

|  |  |  |
| --- | --- | --- |
| Действие | Вес | Количество строк |
| Определение переменной-параметра | 0.3 | 0 |
| Определение временной переменной | 0.5 | 0 |
| Присваивание значения | 0.5 | 0 |
| Вложенное выражение | 0.5 | 0 |
| Сообщение без параметров | 1 | 0 |
| Арифм. операция | 2 | 2 |
| Сообщение с параметрами | 3 | 0 |
| Вызов стандартной функции интерфейса | 5 | 0 |
| Вызов пользовательской функции | 7 | 0 |

Количество описаний сценариев:

,

NKS = 1, т.к. оба класса в той или иной степени непосредственно контактируют с проблемной областью