**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №11**

**Тема: «Перегрузка операций»**

***Цель работы:*** Изучить механизм перегрузки операций.

**Задание №1:** Для одной из прошлых программ создайте статические члены-данные класса:

1. Добавить статическое поле int сount, выступающее в роли счетчика объектов класса.

2. Деструктор класса должен уменьшать на единицу значение счетчика.

3. Добавить статический метод int getCount() возвращающий значение счетчика.

4. Продемонстрировать изменение значения статического поля.

***Ход выполнения***

На листинге 1 представлен код программы. На рисунке 1 – результат теста программы.

Листинг 1 – Код программы

|  |  |
| --- | --- |
| 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.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82.  83.  84.  85.  86.  87.  88.  89.  90.  91.  92.  93.  94.  95.  96.  97.  98.  99.  100.  101.  102.  103.  104.  105.  106.  107.  108.  109.  110.  111.  112.  113.  114.  115.  116.  117.  118.  119.  120.  121.  122.  123.  124.  125.  126.  127.  128.  129.  130.  131.  132.  133.  134.  135.  136.  137. | #include <iostream>  #include <string>  #include <vector>  const bool CUT\_ALL\_SUBSTRINGS = 1; // После того как программа найдёт одну подстроку, она начнёт искать следующую, до тех пор, пока они не кончатся  const bool SHOW\_LEN = 0; // Выводить длинну строки в функции print  using namespace std;  class MyString{  public:  MyString(string str, int len);  MyString(string str);  MyString(vector <char> str);  ~MyString();  vector <char> getStr();  int len();  void print();  MyString operator + (MyString s2);  MyString operator - (MyString s2);  int getCount();  private:  void init(string str, int len);  int \_len = 0;  vector <char> \_str;  static int \_counter;  };  int MyString::\_counter = 0;  void MyString::init(string str, int len){  MyString::\_len = len;  for(int i=0; i<str.length(); i++){  \_str.push\_back(str[i]);  }  \_str.push\_back('\n');  }  MyString::MyString(string str, int len){  if(str.length()>=len){  init(str, len);  }else{  cout << "Ошибка! Переданная строка больше заданного размера!\n";  }  \_counter++;  }  MyString::MyString(string str){  int len = str.length();  init(str, len);  \_counter++;  }  MyString::MyString(vector <char> str){  \_len = str.size()+1;  \_str = str;  \_str.push\_back('\n');  \_counter++;  }  MyString::~MyString(){  \_counter--;  cout << "Удаление объекта\n";  }  vector <char> MyString::getStr(){  return \_str;  }  int MyString::len(){  return \_len;  }  void MyString::print(){  if(SHOW\_LEN){  cout << "Len: " << \_len << endl;  }  for(int i=0; i<=\_len; i++){  cout << MyString::\_str[i];  }  }  MyString MyString::operator + (MyString s2){  vector <char> tmp(\_str);  tmp.pop\_back();  vector <char> str2(s2.getStr());  for(int i=0; i<s2.len(); i++){  tmp.push\_back(str2[i]);  }  tmp.push\_back('\n');  return MyString(tmp);  }  MyString MyString::operator - (MyString s2){  int len = s2.len();  int buff\_len = \_len;  vector <char> buff(\_str);  buff.pop\_back();  if(len<buff\_len){  vector <char> str\_v(s2.getStr());  str\_v.pop\_back();  for(int j=0; j<buff\_len; j++){  if((buff\_len-j)<len){  break;  }  vector <char> tmp;  for(int i=j; i<len+j; i++){  tmp.push\_back(buff[i]);  }  if(tmp == str\_v){  vector <char> buff2;  for(int i=0;i<buff\_len;i++){  if(i>=j && i<(j+len)){  continue;  }  buff2.push\_back(buff[i]);  }  buff.clear();  int new\_len = buff\_len-len;  for(int k=0; k<new\_len; k++){  buff.push\_back(buff2[k]);  }  buff\_len = new\_len;  if(!CUT\_ALL\_SUBSTRINGS){  break;  }  }  }  }  return MyString(buff);  }  int MyString::getCount(){  return \_counter;  }  // --------------------------------------------MAIN---------------------------  int main() {  MyString s (string("str1"), 4);  cout << "Исходная строка "; s.print();  cout << "Объектов класса: " << s.getCount() << endl;  MyString s2 (string("concatination\_con1"));  cout << "Исходная строка "; s2.print();  cout << "Объектов класса: " << s.getCount() << endl;  cout << "После конкатенации "; (s+s2).print();  MyString s3(string("con"));  cout << "Объектов класса: " << s3.getCount() << endl;  MyString s4(s2-s3);  cout << "Вырезанная подстрока "; s4.print();  cout << "Объектов класса: " << s4.getCount() << endl;  return 0;  } |

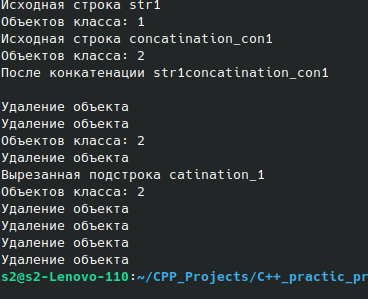


Рисунок 1 – Работа программы

**Задание №2:** Модифицируйте прошлую программу или напишите новую:

1. Определить какие методы являются константными, определить константные параметры и константные возвращаемые значения методов.

2. Добавить константное поле, хранящее идентификатор объекта (номер созданного объекта), предусмотреть методы вывода информации о идентификаторе.

3. Описать и инициализировать обычные и константные объекты.

4. Выполнить вызовы обычных и константных методов для каждого вида объектов.

5. Провести тестирование программы: Откомпилировать программу. Имеются ли ошибки компиляции и какие? Если имеются, то закомментировать соответствующие строки кода и вновь провести компиляцию. Какие предупреждения выдает компилятор и в чем их смысл? Как их можно объяснить с позиции обеспечения надежности программы?

***Ход выполнения***

Для примера, я написал программу, представленную на листинге 2.

Здесь помимо прочего имеется статический атрибут \_counter, который считает кол-во созданных объектов, а также метод getCount, который этот атрибут возвращает.

Также здесь имеется константный метод print, который выводит данные в стандартный вывод.

В функции main я создал два объекта, один константный, а второй обычный, при вызове у константного объекта метода print, ошибки не возникает, а при попытке изменения данных через set, появляется ошибка (рис. 3). При выводе и изменении неконстантного объекта ошибок не возникает, это видно на рисунке 2.

Листинг 2 – Код программы

|  |  |
| --- | --- |
| 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.  60. | #include <iostream>  using namespace std;  class User{  public:  User();  User(char \*ln, char \*n, char \*mn, int a);  ~User();  void print(void) const;  void set(char \*ln, char \*n, char \*mn, int a);  char \*lastname;  char \*name;  char \*middlename;  int age;  int getCount() const;  private:  char login[30];  char password[30];  static int \_counter;  };  int User::\_counter = 0;  User::User(char \*ln, char \*n, char \*mn, int a){  lastname = ln;  middlename = mn;  name = n;  age = a;  \_counter++;  }  User::User(){  lastname = "";  middlename = "";  name = "";  age = 1;  \_counter++;  }  void User::print(void) const{  cout << "ФИО: \n"<<lastname<<endl<<name<<endl<<middlename<<endl<<"Возраст:\n"<<age<<endl;  }  void User::set(char \*ln, char \*n, char \*mn, int a){  lastname = ln;  middlename = mn;  name = n;  age = a;  }  User::~User(void){  cout << "Уничтожение объекта\n";  }  int User::getCount() const{  return \_counter;  }  // -----------------------------------------MAIN---------------------------------  int main(void){  setlocale(LC\_ALL, "ru\_RU.utf8");  const User con\_user("Иванов", "Иван", "Иванович", 43);  cout << "Константный объект:\nprint(): ";con\_user.print();  con\_user.set("Иванов", "Иван", "Иванович", 45);  User ne\_user("Ивасиков", "Василий", "Васильевич", 43);  cout << "Не константный объект:\nprint(): ";ne\_user.print();  ne\_user.set("Ивасиков", "Василий", "Васильевич", 45);  } |

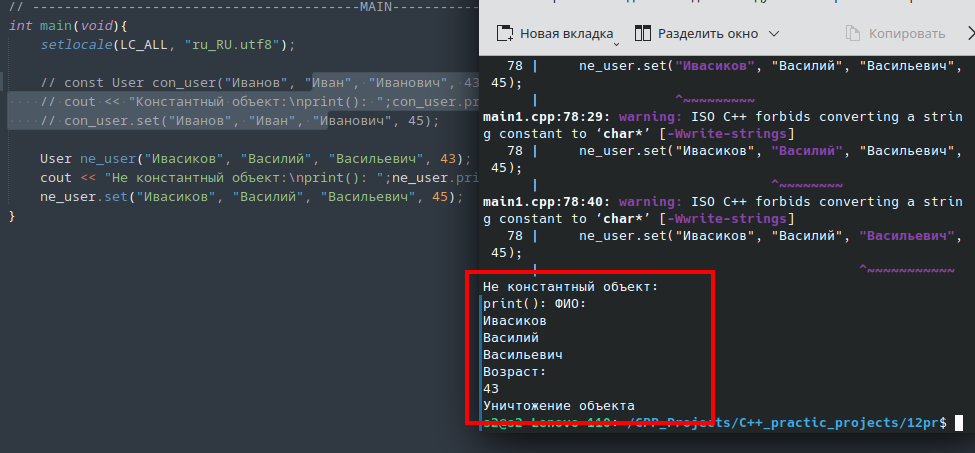


Рисунок 2 – Вывод без ошибок

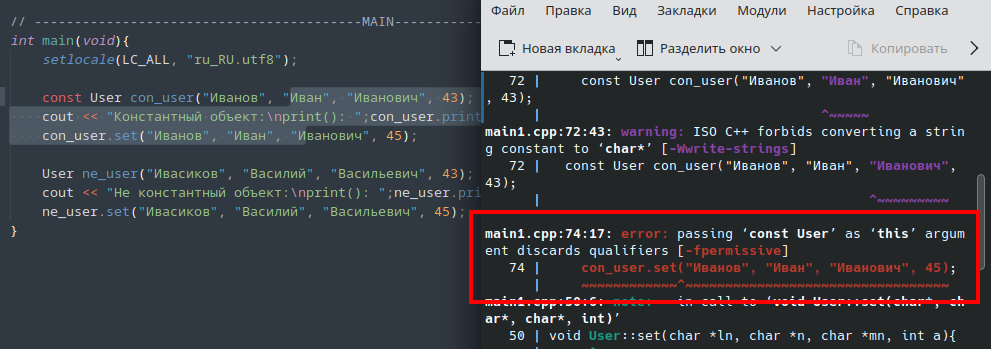


Рисунок 3 – Ошибка

***Задание №3:*** Изучить дополнительный материал по дружественным функциям и напишите небольшую программу, которая их использует.

***Ход выполнения***

Я решил доработать прошлый код и добавил 4 новых функции: setLogin, которая задаёт логин, printPassword, которая печатает пароль, заменяя все символы кроме первого и последнего на \*, getLogin, которая является дружественной и возвращает логин и setPassword, которая также является дружественной и задаёт пароль.

На листинге 3 представлен код программы. На рисунке 4 – результат теста программы. Из результата видно, что дружественные функции могут иметь доступ к приватным атрибутам и даже изменять их.

Листинг 3 – Код программы

|  |  |
| --- | --- |
| 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.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82.  83.  84. | #include <iostream>  #include <string.h>  #include <cstdlib>  using namespace std;  class User{  public:  User();  User(char \*ln, char \*n, char \*mn, int a);  ~User();  void print(void) const;  void set(char \*ln, char \*n, char \*mn, int a);  void setLogin(char \*log){  login = log;  }  void printPassword();  char \*lastname;  char \*name;  char \*middlename;  int age;  int getCount() const;  friend char \*getLogin(User& obj);  friend void setPassword(char \*psw, User& obj);  private:  char \*login;  char \*password;  static int \_counter;  };  int User::\_counter = 0;  User::User(char \*ln, char \*n, char \*mn, int a){  lastname = ln;  middlename = mn;  name = n;  age = a;  \_counter++;  }  User::User(){  lastname = "";  middlename = "";  name = "";  age = 1;  \_counter++;  }  void User::print(void) const{  cout << "ФИО: \n"<<lastname<<endl<<name<<endl<<middlename<<endl<<"Возраст:\n"<<age<<endl;  }  void User::set(char \*ln, char \*n, char \*mn, int a){  lastname = ln;  middlename = mn;  name = n;  age = a;  }  void User::printPassword(){  for (int i = 0; i <= sizeof(password)/sizeof(password[0]); ++i){  if(i==0 or i==(sizeof(password)/sizeof(password[0]))){  cout << password[i];  }else{  cout << '\*';  }  }  cout << endl;  }  User::~User(void){  cout << "Уничтожение объекта\n";  }  int User::getCount() const{  return \_counter;  }  char \*getLogin(User& obj){  return obj.login;  }  void setPassword(char \*psw, User& obj){  obj.password = psw;  }  // -----------------------------------------MAIN---------------------------------  int main(void){  setlocale(LC\_ALL, "ru\_RU.utf8");  User usr("Иванов", "Иван", "Иванович", 43);  usr.setLogin("123\_login");  cout << "Login: " << getLogin(usr);  setPassword("123\_pswrd", usr);  cout << "\nPassword: "; usr.printPassword();  return 0;  } |

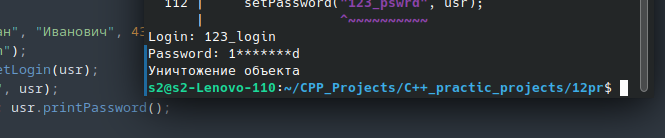


Рисунок 4 – Работа программы