Министерство образования и науки РФ
Государственное образовательное учреждение высшего профессионального образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Кафедра «Информационная безопасность систем и технологий»

ПРОЕКТИРОВАНИЕ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Пояснительная записка к курсовому проекту
по дисциплине «Программно-аппаратные средства обеспечения информационной безопасности»

ПГУ 2.090106.001 ПЗ

Руководитель КР,
доцент, к.т.н. _____Фатеев А. Г.
Исполнитель КР,
студент _____Захаров М. А.

Пенза 2011

РЕФЕРАТ

Отчёт 36 с., 9 рис., 4 источника, 2 прил.

ЗАЩИТА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, НЕСАНКЦИОНИРОВАННОЕ ИСПОЛЬЗОВАНИЕ, ОТЛАДЧИК, ДИЗАССЕМБЛЕР, УТИЛИТА МОНИТОРИНГА, ХЭШ-ФУНКЦИЯ

Объектом исследования являются методы защиты программного обеспечения от несанкционированного копирования.

Цель работы — разработка программной защиты приложения от несанкционированного копирования.

В процессе работы были изучены сведения о защите программ от несанкционированного копирования и методах ограничения функциональности.

В результате исследования была разработана защита программы, генерирующей псевдослучайную последовательность заданной длины, с защитой от несанкционированного копирования методом ограничения времени работы незарегистрированной программы.

СОДЕРЖАНИЕ

«Утверждаю»
Зав. кафедрой ИБСТ
_____С. Л. Зефиров
«____» _____ 2011 г.

ЗАДАНИЕ

на курсовую работу

ВВЕДЕНИЕ

Целью данного курсового проектирования является разработка программной защиты приложения от несанкционированного копирования.

Защита интеллектуальных ресурсов от незаконного использования является на сегодняшний день одним из основных направлений разработки программного обеспечения. Не существует абсолютно надежных методов защиты. Можно утверждать, что достаточно квалифицированные системные программисты, пользующиеся современными средствами анализа работы программного обеспечения (отладчики, дизассемблеры, перехватчики прерываний и т. д.), располагающие достаточным временем, смогут преодолеть практически любую защиту. Поэтому при проектировании системы защиты следует исходить из предположения, что рано или поздно эта защита окажется снятой [1]. Целью проектирования должен быть выбор такого способа защиты, который обеспечит невозможность несанкционированного копирования для заранее определенного круга лиц и в течение ограниченного времени.

В процессе выполнения курсового проекта производится разработка защиты программы-генератора псевдослучайной последовательности от несанкционированного копирования, за счет ограничения времени работы программы. Разработанная защита должна быть исследована с целью анализа стойкости.

1 РАЗРАБОТКА И РЕАЛИЗАЦИЯ АЛГОРИТМА ЗАЩИТЫ

1.1 Описание целей и способа защиты

В соответствии с заданием на курсовой проект, защита программы от несанкционированного копирования осуществляется посредством ограничения времени работы незарегистрированной программы. Данный способ защиты обеспечивает возможность ознакомления пользователя с программой без ее приобретения. Это позволяет предоставить использование программы пользователям, которые не планируют ее постоянно использовать, либо хотят принять решение о целесообразности ее приобретения. Для снятия ограничения по использованию программы следует произвести регистрацию программы с помощью пароля. Знание пароля является свидетельством прав на обладание программой и должно предоставляться правообладателем.

Защита программы должна контролировать наличие регистрации. При этом программа должна регистрироваться однажды и не требовать ввода пароля при каждом запуске. В качестве механизма определения возможностей пользователя при работе с программой выступает признак регистрации программы, сохраняемый в реестре операционной системы. В качестве признака выступает текстовая строка, содержащая ключевую фразу.

Признак регистрации проверяется при запуске. В случае отсутствия регистрации проверяется срок использования незарегистрированной программы. Если он не превышает установленного, программа запускается, в противном случае предлагается зарегистрировать программу. До регистрации программы доступ к полезным функциям программы блокируется.

При регистрации создается соответствующий ключ реестра, в который записывается строка регистрации.

Пароль защищается в коде программы регистрации применением встроенной в Visual Studio хэш-функции.

1.2 Описание алгоритма защиты

При запуске программы проверяется наличие записи в ключе реестра. Если ключ не найден, либо его содержимое не совпадает с признаком регистрации, то осуществляется проверка срока использования программы. Для этого из файла считывается дата установки программы. К полученной дате прибавляется срок использования незарегистрированной программы, равный 5 дням. Полученная дата сравнивается с текущей. Если рассчитанная дата меньше текущей, доступ к полезным функциям программы блокируется и пользователю предлагается произвести регистрацию посредством пароля.

Если пользователь желает пройти регистрацию, то запускается процедура регистрации. Процедура регистрации включает правильность проверки пароля пользователя. Если пароль верен, в ключ реестра записывается строка-признак регистрации. При вводе неверного пароля выдается соответствующее сообщение, и регистрация не проходит. Если содержание ключа реестра, используемого программой, соответствует установленному признаку регистрации, то программа считается зарегистрированной и запускается без проверки срока использования.

Блок-схема разработанного алгоритма приведена в Приложении А.

1.3 Описание программной реализации алгоритма защиты

Для реализации разработанного алгоритма защиты программы была разработана программа, выполняющая генерацию псевдослучайной последовательности со встроенной защитой от несанкционированного копирования, а также программа установки, осуществляющая настройку параметров защиты для основной программы.

Для разработки программы использовалась интегрированная среда разработки Visual Studio Express 2010. Для написания программы был использован язык программирования Visual C++. Функция регистрации позволяет снять ограничения для легальных пользователей. Она включает проверку пароля пользователя и, в случае ввода правильного пароля, создает ключ реестра, содержащий признак регистрации. Данная функция реализована в виде обработчика нажатия кнопки на главной форме проекта (листинг 1).

Листинг 1: Функция регистрации

```
1  private: System::Void button2_Click(System::Object^  sender,
       System::EventArgs^  e) {
2   int pass = this->textBox1->Text->GetHashCode();
3   int right_pass = 1364505728;
4   if(pass==right_pass)
5   {
6             RegistryKey^ rk = nullptr;
7             rk = Registry::CurrentUser->OpenSubKey("GPSP",true);
8             if (rk==nullptr)
9             {
10                            RegistryKey^ rk = Registry::
      CurrentUser->CreateSubKey("GPSP");
11                            rk->SetValue("line","forza");
12                            rk->Close();
13            }
14            else
15            {
```

```
16                          Registry::CurrentUser->DeleteSubKey("
    GPSP");
17                          RegistryKey^ rk = Registry::
    CurrentUser->CreateSubKey("GPSP");
18                          rk->SetValue("line","forza");
19                          rk->Close();
20              }
21              this->groupBox1->Hide();
22              this->groupBox2->Show();
23              MessageBox::Show("Зарегистрировано");
24  }
25  else
26  {
27          MessageBox::Show("Неправильный пароль");
28  }
29
30  }
```

Основная программа содержит функцию проверки наличия ключа реестра и правильность значения, записанного в него.

Данная функция вызывается при загрузке программы, при обработке события `Form_Load`. Если регистрация отсутствует, программа проверяет, не превышен ли срок использования программы. Для этого из файла созданного при установке программы считывается дата установки. Сравнение дат, осуществляется с использованием методов встроенного класса `DateTime` (листинг 2) [2].

Листинг 2: Проверка признака регистрации

```
1   bool reg = true;
2   RegistryKey^ rk = nullptr;
3   rk = Registry::CurrentUser->OpenSubKey("GPSP",true);
4   if (rk==nullptr)
5   {
6           reg = false;
7   }
8   else
9   {
10          String^ value = rk->GetValue("line")->ToString();
```

```
11          if(value!="forza") reg = false;
12  }
13  DateTime^ dt = DateTime::Now;
14  if(!reg)
15  {
16
17          StreamReader^ read_date = gcnew StreamReader("param");
18          DateTime^ start = Convert::ToDateTime(read_date->
      ReadLine());
19          if(DateTime::Compare(start->AddDays(5), dt->Date)<0)
20          {
21                  MessageBox::Show("Срок использования
      незарегистрированной версии истек. Хотите зарегистрировать
      программу");
22                  this->groupBox2->Hide();
23          }
24          else this->groupBox1->Hide();
25  }
26  else this->groupBox1->Hide();
```

Интерфейс основной полезной программы приведен на рисунке 1.

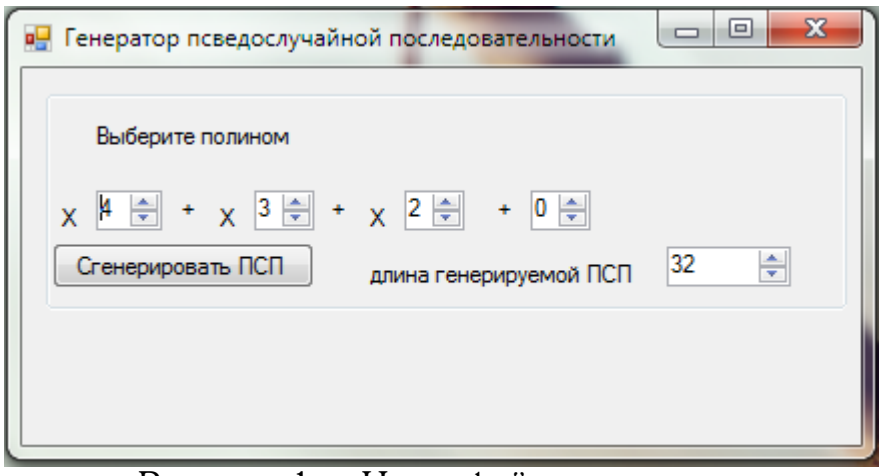

Рисунок 1 – Интерфейс программы

Для проверки корректности работы защиты программы было осуществлено изменение системной даты на 7 дней вперед. На рисунке 2 отображено сообщение об окончании периода использования незарегистрированной программы.

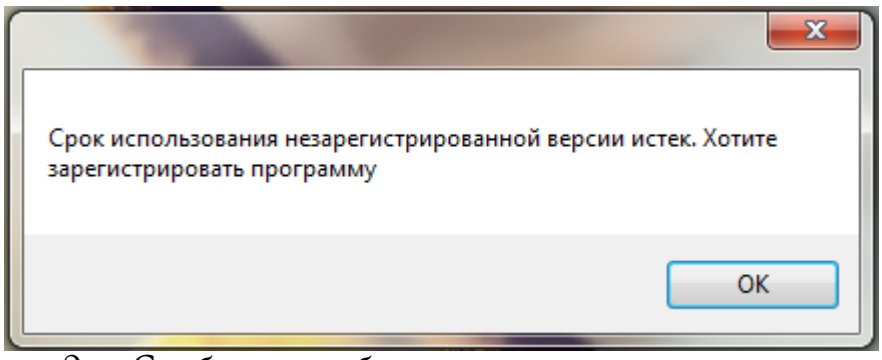

Рисунок 2 – Сообщение об окончании периода использования незарегистрированной программы

Блокирование полезных функций программы после окончания периода использования незарегистрированной программы представлено на рисунке 3.

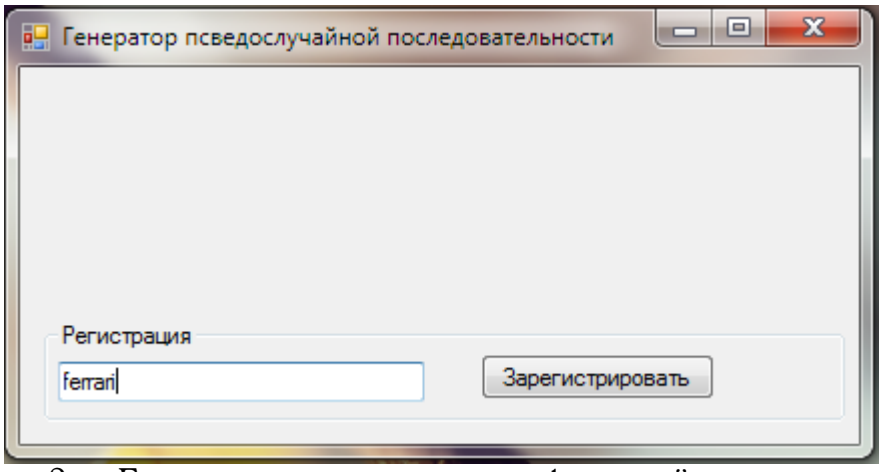

Рисунок 3 – Блокирование полезных функций после окончания периода использования незарегистрированной программы

Полный текст основной программы, а также программы установки приведен в Приложении Б.

2 АНАЛИЗ СТОЙКОСТИ ИСПОЛЬЗОВАННОГО АЛГОРИТМА ЗАЩИТЫ

В ходе исследования выявляются сведения, которые могут быть использованы нарушителем для проведения атак на защиту программы. На данном этапе использовались следующие программные средства анализа[4]:

- дизассемблер *IDA*, позволяющий произвести статическое исследование дизассемблированного листинга программы;
- утилита мониторинга *ProcessMonitor*, осуществляющая контроль активности приложения при обращениях к ресурсам файловой системы, реестра и системным вызовам;
- отладчик *OllyDbg*, позволяющий произвести динамическое исследование логики работы программы.

2.1 Анализ защиты с использованием дизассемблера

Первым этапом исследования являлось дизассемблирование программы. В результате анализа полученного кода выявлен механизм проверки регистрации, представленный на рисунке 4.

На данном рисунке отмечены функции считывания значения ключа реестра и функция сравнения с признаком регистрации. Признак регистрации при этом хранится в явном виде (строка). Данные сведения могут быть использованы при создании признака регистрации несанкционированным способом.

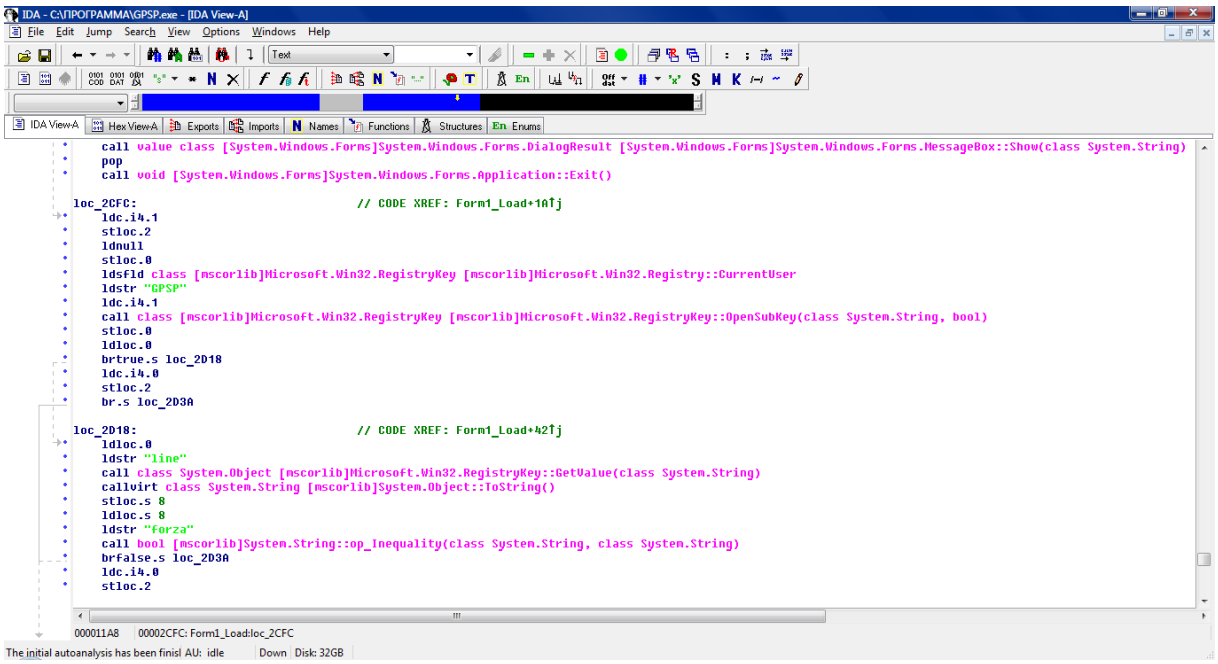

Рисунок 4 – Функция проверки регистрации программы

В обработчике кнопки регистрации были выявлены механизмы проверки пароля пользователя и хэш от верного пароля, а также создания признака регистрации в ключе реестра. Данные элементы отмечены на рисунке 5.

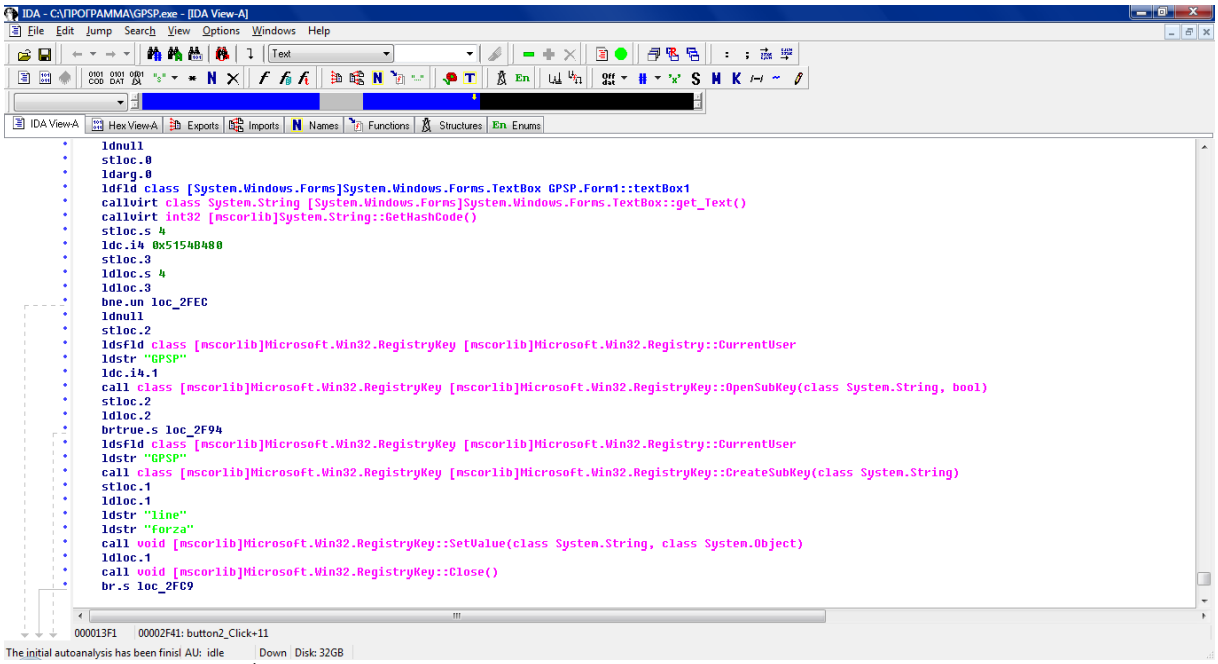

Рисунок 5 – Проверка пароля пользователя при регистрации

Таким образом, атака на пароль затруднена, поскольку в про-

грамме хранится лишь проверочное значение хэш, и наиболее эффективным способом будет простой перебор всех возможных паролей.

Также была исследована функция проверки срока использования программы. Данная функция представлена на рисунке 6.

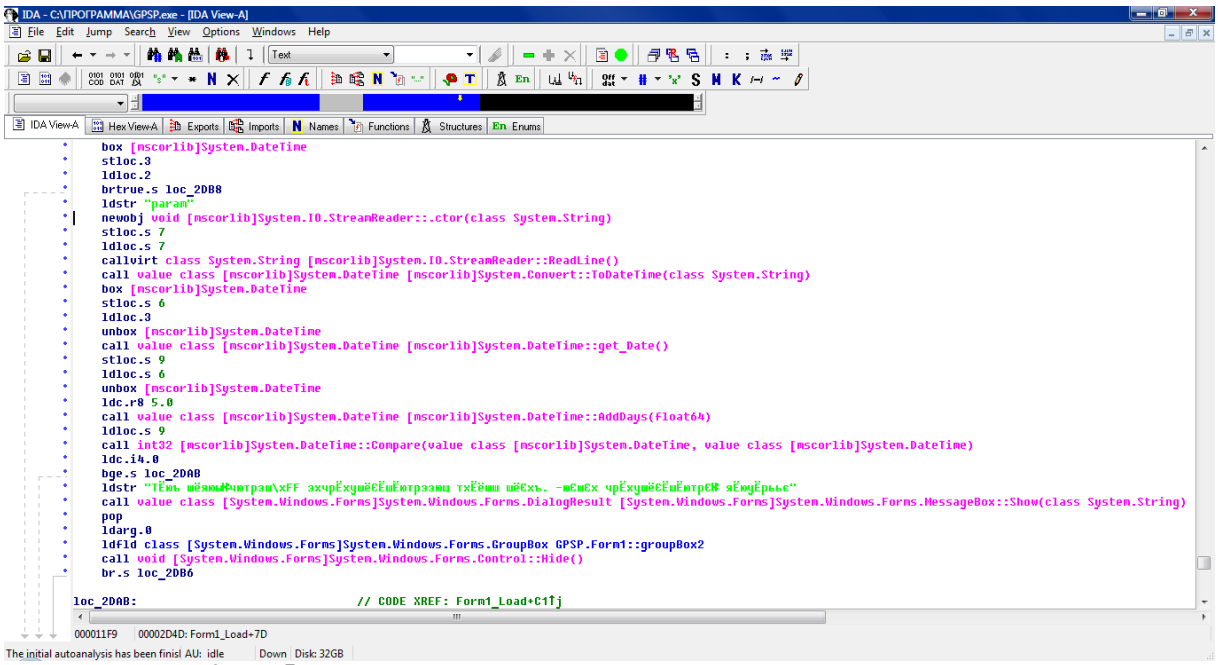

Рисунок 6 – Функция проверки срока использования программы

На рисунке 6 отмечено считывание даты установки из файла `param`, а также срок равный 5 дням и функция сравнения дат. Данные сведения могут быть использованы при попытках модификации даты установки, либо при модификации программы.

2.2 Анализ защиты с использованием монитора процессов

Далее была проанализированы обращения программы к ресурсам файловой системы и реестра при запуске и проверке регистрации, с помощью программы *ProcessMonitor*. На рисунке 7 представлено считывание признака регистрации из ключа реестра при запуске.

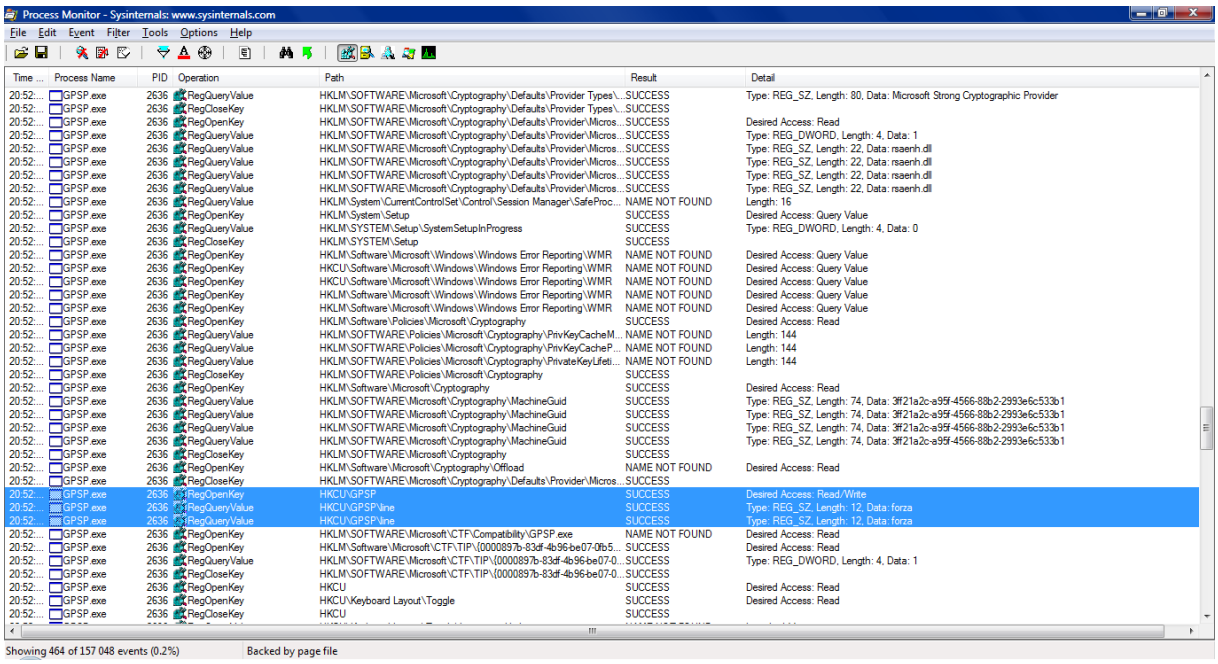

Рисунок 7 – Обращения к реестру для проверки признака регистрации

Из полученных сведений можно построить алгоритм работы механизма проверки регистрации. Кроме того, можно установить содержимое ключа регистрации программы (строка-признак регистрации `forza` отображается как содержимое ключа). Данные сведения могут быть использованы при создании признака регистрации несанкционированным способом.

2.3 Анализ защиты с использованием отладчика

В ходе исследования средствами отладчика выявлен условный переход, управляющий проверкой регистрации. На рисунке 8 на него установлена точка останова.

Модификация данного условного перехода средствами отладчика (например, отключение проверки заменой перехода на команду

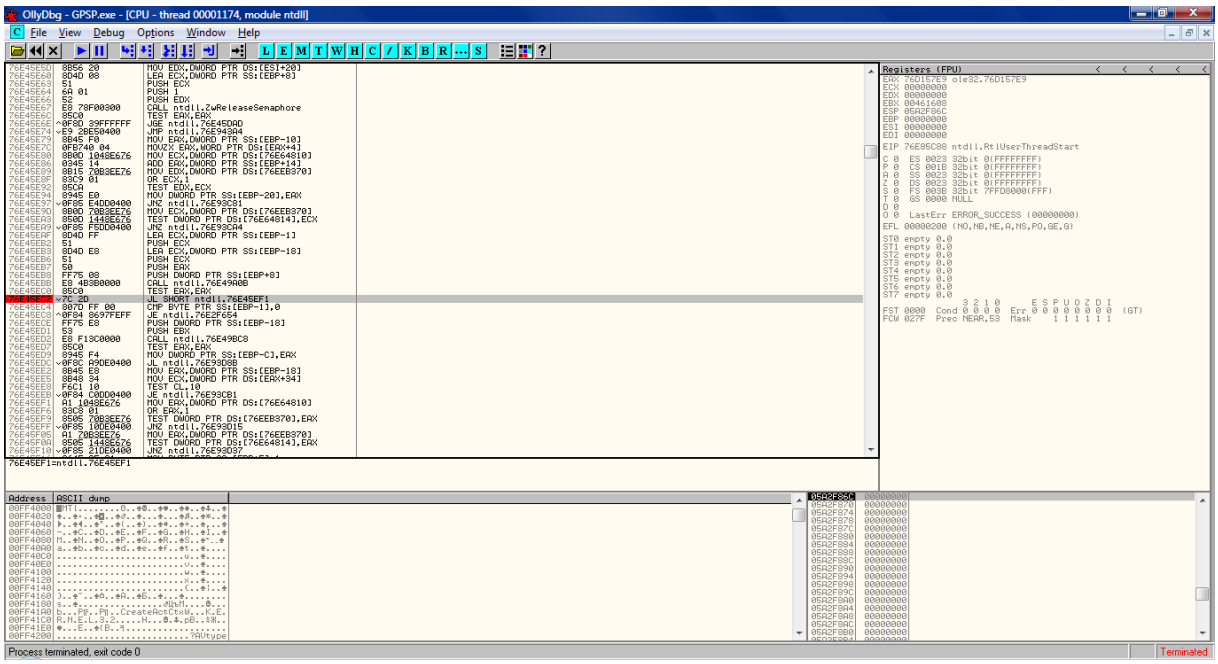

Рисунок 8 – Условный переход, управляющий проверкой регистрации

NOP) приведет к снятию защиты программы (результаты проверки не будут обрабатываться для управления загрузкой программы).

В ходе исследования были выявлены уязвимости, позволяющие обойти защиту программы: возможность изучения механизмов регистрации и проверки регистрации, незащищенное хранение признака регистрации, возможность модификации программы с целью отключения защиты.

Для повышения защиты программы следует ввести следующие усиления [4]:

– разделение признака регистрации на несколько файлов и их шифрование с использованием данных даты установки, для вычисления ключа;
– использование самомодифицирующегося кода: шифрование критических участков программы и расшифровка их непосредственно в память процесса;
– использование средств защиты от отладки;
– защита файла, хранящего дату установки с использованием шифрования;
– усложнения алгоритма работы программы с помощью лож-

ных ветвей алгоритма и функций, не влияющих на работу программы;

– отказ от использования осмысленных имен функций и использование функций переходников.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были изучены сведения о защите программ от несанкционированного копирования и методах ограничения функциональности. В результате выполнения курсового проекта разработана защита программы, генерирующей псевдослучайную последовательность заданной длины, с защитой от несанкционированного копирования методом ограничения времени работы незарегистрированной программы.

Для разработанной защиты был проведен анализ стойкости и сформированы предложения по устранению выявленных недостатков.

Таким образом, требования, описанные в техническом задании на данный курсовой проект, были выполнены полностью.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Александр Фролов, Григорий Фролов. MS-DOS для программиста М.: Диалог-МИФИ, 1995, 253 стр.
2. Самоучитель хакера: Подробное иллюстрированное руководство: Alex Atsctoy. — М.: «Лучшие книги», 2005. — 192 с.
3. Взлом программного обеспечения: анализ и использование кода: Перевод с английского — М.: Издательский дом «Вильямс», 2005. — 400 с.
4. Защита от взлома: Пер. с англ. Слинкина А. А. — М.: Издательский Дом ДМК-пресс, 2006. — 784 с.

ПРИЛОЖЕНИЕ А

(справочное)

Алгоритм защиты программы

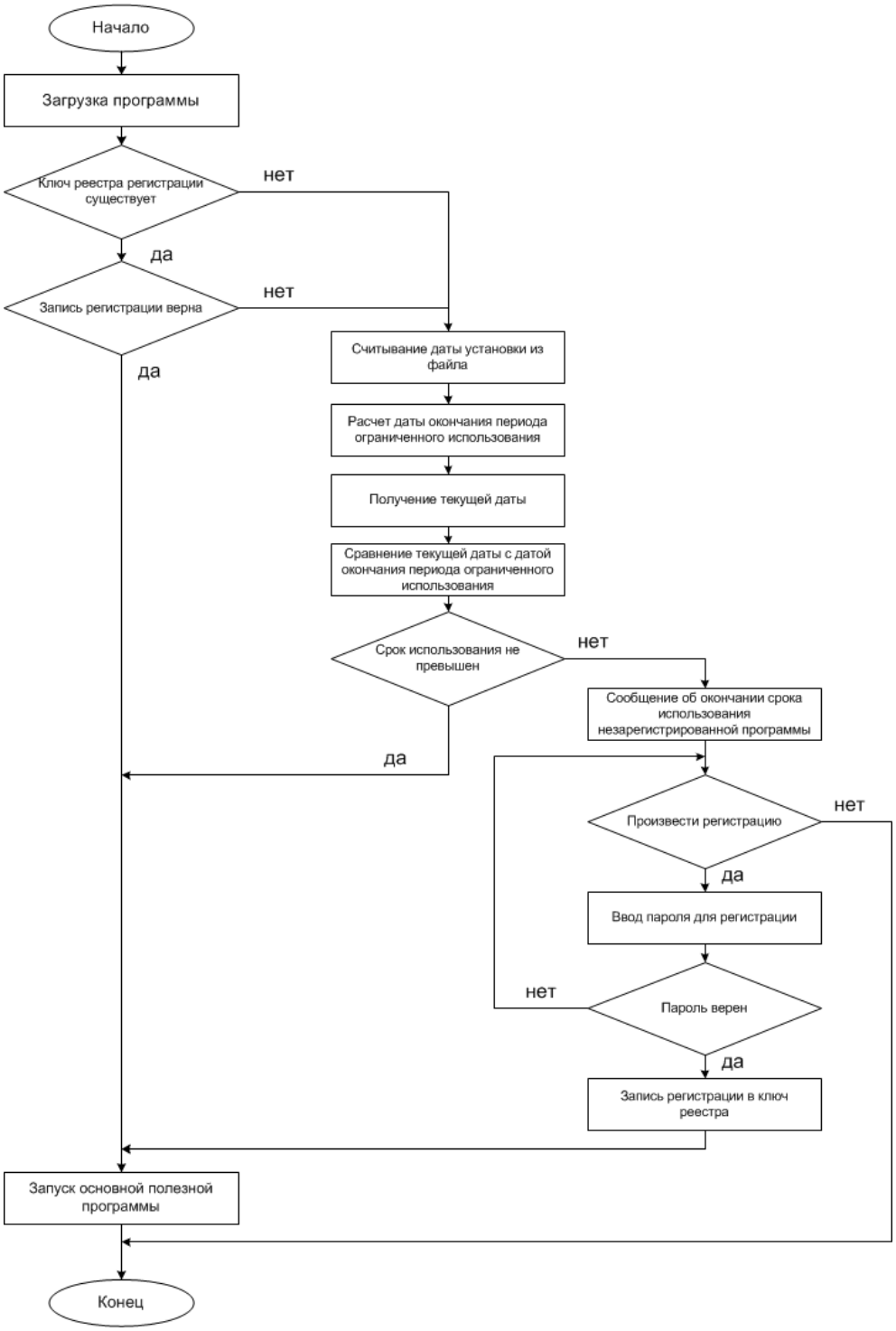

Рисунок А.1 — Алгоритм защиты программы

ПРИЛОЖЕНИЕ Б

(справочное)

Листинги разработанных программ

Листинг 3: ПРОГРАММА УСТАНОВКИ

```
1
2  #pragma once
3
4
5  namespace GPSP_install {
6
7  using namespace System;
8  using namespace System::ComponentModel;
9  using namespace System::Collections;
10 using namespace System::Windows::Forms;
11 using namespace System::Data;
12 using namespace System::Drawing;
13 using namespace System::IO;
14
15 /// <summary>
16 /// Сводка для Form1
17 ///
18 /// Внимание! При изменении имени этого класса необходимо
      также изменить
19 ///          свойство имени файла ресурсов ("Resource File Name
      ") для средства компиляции управляемого ресурса,
20 ///          связанного со всеми файлами с расширением .resx,
      от которых зависит данный класс. В противном случае,
21 ///          конструкторы не смогут правильно работать с
      локализованными
22 ///          ресурсами, сопоставленными данной форме.
23 /// </summary>
24 public ref class Form1 : public System::Windows::Forms::Form
25 {
26 public:
27 Form1(void)
28 {
29         InitializeComponent();
```

```
30          //
31          //TODO: добавьте код конструктора
32          //
33  }
34
35  protected:
36  /// <summary>
37  /// Освободить все используемые ресурсы.
38  /// </summary>
39  ~Form1()
40  {
41          if (components)
42          {
43                  delete components;
44          }
45  }
46  private: System::Windows::Forms::Button^  button1;
47  protected:
48
49  private:
50  /// <summary>
51  /// Требуется переменная конструктора.
52  /// </summary>
53  System::ComponentModel::Container ^components;
54
55  #pragma region Windows Form Designer generated code
56  /// <summary>
57  /// Обязательный метод для поддержки конструктора - не
        изменяйте
58  /// содержимое данного метода при помощи редактора кода.
59  /// </summary>
60  void InitializeComponent(void)
61  {
62          this->button1 = (gcnew System::Windows::Forms::Button()
        );
63          this->SuspendLayout();
64          //
65          // button1
66          //
```

```
67          this->button1->Location = System::Drawing::Point(12,
    22);
68          this->button1->Name = L"button1";
69          this->button1->Size = System::Drawing::Size(75, 23);
70          this->button1->TabIndex = 0;
71          this->button1->Text = L"Установить";
72          this->button1->UseVisualStyleBackColor = true;
73          this->button1->Click += gcnew System::EventHandler(this
    , &Form1::button1_Click);
74          //
75          // Form1
76          //
77          this->AutoScaleDimensions = System::Drawing::SizeF(6,
    13);
78          this->AutoScaleMode = System::Windows::Forms::
    AutoScaleMode::Font;
79          this->ClientSize = System::Drawing::Size(198, 58);
80          this->Controls->Add(this->button1);
81          this->Name = L"Form1";
82          this->Text = L"Установка";
83          this->ResumeLayout(false);
84
85  }
86  #pragma endregion
87  private: System::Void button1_Click(System::Object^  sender,
    System::EventArgs^  e) {
88          FolderBrowserDialog^ folder_dialog = gcnew
    FolderBrowserDialog();
89          folder_dialog->ShowDialog();
90          if(folder_dialog->SelectedPath != "")
91          {
92                  StreamWriter^ write = gcnew StreamWriter(
    folder_dialog->SelectedPath + "\\param");
93                  DateTime^ dt = DateTime::Now;
94                  write->WriteLine(dt->ToLongDateString());
95                  write->Close();
96                  File::Copy("GPSP.exe", folder_dialog->
    SelectedPath + "\\GPSP.exe");
97                  MessageBox::Show("Установлено");
```

```
98              }
99      }
100 };
101 }
```

Листинг 4: ОСНОВНАЯ ПОЛЕЗНАЯ ПРОГРАММА

```
1  #pragma once
2  namespace GPSP {
3
4  using namespace System;
5  using namespace System::ComponentModel;
6  using namespace System::Collections;
7  using namespace System::Windows::Forms;
8  using namespace System::Data;
9  using namespace System::Drawing;
10 using namespace System::IO;
11 using namespace Microsoft::Win32;
12
13
14
15 /// <summary>
16 /// Сводка для Form1
17 ///
18 /// Внимание! При изменении имени этого класса необходимо
      также изменить
19 ///          свойство имени файла ресурсов ("Resource File Name
      ") для средства компиляции управляемого ресурса,
20 ///          связанного со всеми файлами с расширением .resx,
      от которых зависит данный класс. В противном случае,
21 ///          конструкторы не смогут правильно работать с
      локализованными
22 ///          ресурсами, сопоставленными данной форме.
23 /// </summary>
24 public ref class Form1 : public System::Windows::Forms::Form
25 {
26 public:
27         Form1(void)
28         {
29                 InitializeComponent();
```

```
30                  //
31                  //TODO: добавьте код конструктора
32                  //
33          }
34
35  protected:
36          /// <summary>
37          /// Освободить все используемые ресурсы.
38          /// </summary>
39          ~Form1()
40          {
41                  if (components)
42                  {
43                          delete components;
44                  }
45          }
46  private: System::Windows::Forms::Label^  label1;
47  protected:
48  private: System::Windows::Forms::NumericUpDown^  numericUpDown1
        ;
49  private: System::Windows::Forms::Label^  label2;
50  private: System::Windows::Forms::Label^  label3;
51  private: System::Windows::Forms::NumericUpDown^  numericUpDown2
        ;
52  private: System::Windows::Forms::Label^  label4;
53  private: System::Windows::Forms::Label^  label5;
54  private: System::Windows::Forms::NumericUpDown^  numericUpDown3
        ;
55  private: System::Windows::Forms::Label^  label6;
56  private: System::Windows::Forms::NumericUpDown^  numericUpDown4
        ;
57  private: System::Windows::Forms::Label^  label8;
58  private: System::Windows::Forms::Button^  button1;
59  private: System::Windows::Forms::NumericUpDown^  numericUpDown5
        ;
60  private: System::Windows::Forms::Label^  label7;
61  private: System::Windows::Forms::GroupBox^  groupBox1;
62  private: System::Windows::Forms::Button^  button2;
63  private: System::Windows::Forms::TextBox^  textBox1;
```

```
64 private: System::Windows::Forms::GroupBox^  groupBox2;
65
66 private:
67         /// <summary>
68         /// Требуется переменная конструктора.
69         /// </summary>
70         System::ComponentModel::Container ^components;
71
72 #pragma region Windows Form Designer generated code
73 /// <summary>
74 /// Обязательный метод для поддержки конструктора - не
      изменяйте
75 /// содержимое данного метода при помощи редактора кода.
76 /// </summary>
77 void InitializeComponent(void)
78 {
79 this->label1 = (gcnew System::Windows::Forms::Label());
80 this->numericUpDown1 = (gcnew System::Windows::Forms::
      NumericUpDown());
81 this->label2 = (gcnew System::Windows::Forms::Label());
82 this->label3 = (gcnew System::Windows::Forms::Label());
83 this->numericUpDown2 = (gcnew System::Windows::Forms::
      NumericUpDown());
84 this->label4 = (gcnew System::Windows::Forms::Label());
85 this->label5 = (gcnew System::Windows::Forms::Label());
86 this->numericUpDown3 = (gcnew System::Windows::Forms::
      NumericUpDown());
87 this->label6 = (gcnew System::Windows::Forms::Label());
88 this->numericUpDown4 = (gcnew System::Windows::Forms::
      NumericUpDown());
89 this->label8 = (gcnew System::Windows::Forms::Label());
90 this->button1 = (gcnew System::Windows::Forms::Button());
91 this->numericUpDown5 = (gcnew System::Windows::Forms::
      NumericUpDown());
92 this->label7 = (gcnew System::Windows::Forms::Label());
93 this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
94 this->button2 = (gcnew System::Windows::Forms::Button());
95 this->textBox1 = (gcnew System::Windows::Forms::TextBox());
96 this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
```

```
97  (cli::safe_cast<System::ComponentModel::ISupportInitialize^ >(
       this->numericUpDown1))->BeginInit();
98  (cli::safe_cast<System::ComponentModel::ISupportInitialize^ >(
       this->numericUpDown2))->BeginInit();
99  (cli::safe_cast<System::ComponentModel::ISupportInitialize^ >(
       this->numericUpDown3))->BeginInit();
100 (cli::safe_cast<System::ComponentModel::ISupportInitialize^ >(
       this->numericUpDown4))->BeginInit();
101 (cli::safe_cast<System::ComponentModel::ISupportInitialize^ >(
       this->numericUpDown5))->BeginInit();
102 this->groupBox1->SuspendLayout();
103 this->groupBox2->SuspendLayout();
104 this->SuspendLayout();
105 //
106 // label1
107 //
108 this->label1->AutoSize = true;
109 this->label1->Location = System::Drawing::Point(22, 20);
110 this->label1->Name = L"label1";
111 this->label1->Size = System::Drawing::Size(104, 13);
112 this->label1->TabIndex = 0;
113 this->label1->Text = L"Выберите полином";
114 //
115 // numericUpDown1
116 //
117 this->numericUpDown1->Location = System::Drawing::Point(25, 54)
       ;
118 this->numericUpDown1->Maximum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {31, 0, 0, 0});
119 this->numericUpDown1->Minimum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {4, 0, 0, 0});
120 this->numericUpDown1->Name = L"numericUpDown1";
121 this->numericUpDown1->Size = System::Drawing::Size(33, 20);
122 this->numericUpDown1->TabIndex = 1;
123 this->numericUpDown1->Value = System::Decimal(gcnew cli::array<
        System::Int32 >(4) {4, 0, 0, 0});
124 //
125 // label2
126 //
```

```
127 this->label2->AutoSize = true;
128 this->label2->Location = System::Drawing::Point(5, 61);
129 this->label2->Name = L"label2";
130 this->label2->Size = System::Drawing::Size(14, 13);
131 this->label2->TabIndex = 2;
132 this->label2->Text = L"X";
133 //
134 // label3
135 //
136 this->label3->AutoSize = true;
137 this->label3->Location = System::Drawing::Point(65, 56);
138 this->label3->Name = L"label3";
139 this->label3->Size = System::Drawing::Size(13, 13);
140 this->label3->TabIndex = 3;
141 this->label3->Text = L"+";
142 //
143 // numericUpDown2
144 //
145 this->numericUpDown2->Location = System::Drawing::Point(104,
      54);
146 this->numericUpDown2->Maximum = System::Decimal(gcnew cli::
      array< System::Int32 >(4) {31, 0, 0, 0});
147 this->numericUpDown2->Minimum = System::Decimal(gcnew cli::
      array< System::Int32 >(4) {3, 0, 0, 0});
148 this->numericUpDown2->Name = L"numericUpDown2";
149 this->numericUpDown2->Size = System::Drawing::Size(30, 20);
150 this->numericUpDown2->TabIndex = 4;
151 this->numericUpDown2->Value = System::Decimal(gcnew cli::array<
      System::Int32 >(4) {3, 0, 0, 0});
152 //
153 // label4
154 //
155 this->label4->AutoSize = true;
156 this->label4->Location = System::Drawing::Point(84, 61);
157 this->label4->Name = L"label4";
158 this->label4->Size = System::Drawing::Size(14, 13);
159 this->label4->TabIndex = 5;
160 this->label4->Text = L"X";
161 //
```

```
162 // label5
163 //
164 this->label5->AutoSize = true;
165 this->label5->Location = System::Drawing::Point(159, 61);
166 this->label5->Name = L"label5";
167 this->label5->Size = System::Drawing::Size(14, 13);
168 this->label5->TabIndex = 8;
169 this->label5->Text = L"X";
170 //
171 // numericUpDown3
172 //
173 this->numericUpDown3->Location = System::Drawing::Point(179,
       54);
174 this->numericUpDown3->Maximum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {31, 0, 0, 0});
175 this->numericUpDown3->Minimum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {2, 0, 0, 0});
176 this->numericUpDown3->Name = L"numericUpDown3";
177 this->numericUpDown3->Size = System::Drawing::Size(30, 20);
178 this->numericUpDown3->TabIndex = 7;
179 this->numericUpDown3->Value = System::Decimal(gcnew cli::array<
        System::Int32 >(4) {2, 0, 0, 0});
180 //
181 // label6
182 //
183 this->label6->AutoSize = true;
184 this->label6->Location = System::Drawing::Point(140, 56);
185 this->label6->Name = L"label6";
186 this->label6->Size = System::Drawing::Size(13, 13);
187 this->label6->TabIndex = 6;
188 this->label6->Text = L"+";
189 //
190 // numericUpDown4
191 //
192 this->numericUpDown4->Location = System::Drawing::Point(242,
       54);
193 this->numericUpDown4->Maximum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {1, 0, 0, 0});
194 this->numericUpDown4->Name = L"numericUpDown4";
```

```
195 this->numericUpDown4->Size = System::Drawing::Size(30, 20);
196 this->numericUpDown4->TabIndex = 10;
197 //
198 // label8
199 //
200 this->label8->AutoSize = true;
201 this->label8->Location = System::Drawing::Point(223, 56);
202 this->label8->Name = L"label8";
203 this->label8->Size = System::Drawing::Size(13, 13);
204 this->label8->TabIndex = 9;
205 this->label8->Text = L"+";
206 //
207 // button1
208 //
209 this->button1->Location = System::Drawing::Point(3, 80);
210 this->button1->Name = L"button1";
211 this->button1->Size = System::Drawing::Size(131, 23);
212 this->button1->TabIndex = 11;
213 this->button1->Text = L"Сгенерировать ПСП";
214 this->button1->UseVisualStyleBackColor = true;
215 this->button1->Click += gcnew System::EventHandler(this, &Form1
       ::button1_Click);
216 //
217 // numericUpDown5
218 //
219 this->numericUpDown5->Location = System::Drawing::Point(310,
       82);
220 this->numericUpDown5->Maximum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {1000, 0, 0, 0});
221 this->numericUpDown5->Minimum = System::Decimal(gcnew cli::
       array< System::Int32 >(4) {32, 0, 0, 0});
222 this->numericUpDown5->Name = L"numericUpDown5";
223 this->numericUpDown5->Size = System::Drawing::Size(62, 20);
224 this->numericUpDown5->TabIndex = 12;
225 this->numericUpDown5->Value = System::Decimal(gcnew cli::array<
        System::Int32 >(4) {32, 0, 0, 0});
226 //
227 // label7
228 //
```

```
229 this->label7->AutoSize = true;
230 this->label7->Location = System::Drawing::Point(159, 89);
231 this->label7->Name = L"label7";
232 this->label7->Size = System::Drawing::Size(138, 13);
233 this->label7->TabIndex = 13;
234 this->label7->Text = L"длина генерируемой ПСП";
235 //
236 // groupBox1
237 //
238 this->groupBox1->Controls->Add(this->button2);
239 this->groupBox1->Controls->Add(this->textBox1);
240 this->groupBox1->Location = System::Drawing::Point(12, 127);
241 this->groupBox1->Name = L"groupBox1";
242 this->groupBox1->Size = System::Drawing::Size(400, 50);
243 this->groupBox1->TabIndex = 14;
244 this->groupBox1->TabStop = false;
245 this->groupBox1->Text = L"Регистрация";
246 //
247 // button2
248 //
249 this->button2->Location = System::Drawing::Point(218, 16);
250 this->button2->Name = L"button2";
251 this->button2->Size = System::Drawing::Size(117, 23);
252 this->button2->TabIndex = 1;
253 this->button2->Text = L"Зарегистрировать";
254 this->button2->UseVisualStyleBackColor = true;
255 this->button2->Click += gcnew System::EventHandler(this, &Form1
       ::button2_Click);
256 //
257 // textBox1
258 //
259 this->textBox1->Location = System::Drawing::Point(7, 20);
260 this->textBox1->Name = L"textBox1";
261 this->textBox1->Size = System::Drawing::Size(183, 20);
262 this->textBox1->TabIndex = 0;
263 //
264 // groupBox2
265 //
266 this->groupBox2->Controls->Add(this->label1);
```

```
267 this->groupBox2->Controls->Add(this->numericUpDown1);
268 this->groupBox2->Controls->Add(this->label7);
269 this->groupBox2->Controls->Add(this->label2);
270 this->groupBox2->Controls->Add(this->numericUpDown5);
271 this->groupBox2->Controls->Add(this->label3);
272 this->groupBox2->Controls->Add(this->button1);
273 this->groupBox2->Controls->Add(this->numericUpDown2);
274 this->groupBox2->Controls->Add(this->numericUpDown4);
275 this->groupBox2->Controls->Add(this->label4);
276 this->groupBox2->Controls->Add(this->label8);
277 this->groupBox2->Controls->Add(this->label6);
278 this->groupBox2->Controls->Add(this->label5);
279 this->groupBox2->Controls->Add(this->numericUpDown3);
280 this->groupBox2->Location = System::Drawing::Point(12, 7);
281 this->groupBox2->Name = L"groupBox2";
282 this->groupBox2->Size = System::Drawing::Size(400, 114);
283 this->groupBox2->TabIndex = 15;
284 this->groupBox2->TabStop = false;
285 //
286 // Form1
287 //
288 this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
289 this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::
       Font;
290 this->ClientSize = System::Drawing::Size(419, 188);
291 this->Controls->Add(this->groupBox2);
292 this->Controls->Add(this->groupBox1);
293 this->Name = L"Form1";
294 this->Text = L"Генератор псведослучайной последовательности";
295 this->Load += gcnew System::EventHandler(this, &Form1::
       Form1_Load);
296 (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(
       this->numericUpDown1))->EndInit();
297 (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(
       this->numericUpDown2))->EndInit();
298 (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(
       this->numericUpDown3))->EndInit();
299 (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(
       this->numericUpDown4))->EndInit();
```

```
300 (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(
      this->numericUpDown5))->EndInit();
301 this->groupBox1->ResumeLayout(false);
302 this->groupBox1->PerformLayout();
303 this->groupBox2->ResumeLayout(false);
304 this->groupBox2->PerformLayout();
305 this->ResumeLayout(false);
306
307 }
308 #pragma endregion
309
310         unsigned long line;
311
312 private: System::Void Form1_Load(System::Object^  sender,
      System::EventArgs^  e) {
313  if(!File::Exists("param"))
314  {
315          MessageBox::Show("Ошибка");
316          Application::Exit();
317  }
318  bool reg = true;
319  RegistryKey^ rk = nullptr;
320  rk = Registry::CurrentUser->OpenSubKey("GPSP",true);
321  if (rk==nullptr)
322  {
323         reg = false;
324  }
325  else
326  {
327          String^ value = rk->GetValue("line")->ToString();
328          if(value!="forza") reg = false;
329  }
330  DateTime^ dt = DateTime::Now;
331  if(!reg)
332  {
333
334          StreamReader^ read_date = gcnew StreamReader("param");
335          DateTime^ start = Convert::ToDateTime(read_date->
     ReadLine());
```

```
336             if(DateTime::Compare(start->AddDays(5), dt->Date)<0)
337             {
338                     MessageBox::Show("Срок использования
    незарегистрированной версии истек. Хотите зарегистрировать
    программу");
339                     this->groupBox2->Hide();
340             }
341             else this->groupBox1->Hide();
342  }
343  else this->groupBox1->Hide();
344
345  Random^ rnd = gcnew Random(dt->ToBinary());
346  line = line & 0x00;
347  unsigned long cur_bit = 0;
348  for(int i = 0; i < 31; i++)
349  {
350             cur_bit = rnd->Next()%2;
351             line = (line << 1) ^ (cur_bit & 0x01);
352  }
353
354
355                     }
356 private: System::Void button1_Click(System::Object^  sender,
    System::EventArgs^  e) {
357 int first = Convert::ToInt32(this->numericUpDown1->Value);
358 int second = Convert::ToInt32(this->numericUpDown2->Value);
359 int third = Convert::ToInt32(this->numericUpDown3->Value);
360 int free = Convert::ToInt32(this->numericUpDown4->Value);
361 int length = Convert::ToInt32(this->numericUpDown5->Value);
362 unsigned long in_file = 0;
363 Stream^ str = File::Open("PSP", FileMode::OpenOrCreate,
    FileAccess::Write);
364 BinaryWriter^ bwr = gcnew BinaryWriter(str);
365 for(int i = 0; i < length; i++)
366 {
367         for(int j = 0; j < 31; j++)
368         {
369                 line = (line << 1) ^ ((((line >> (first-1)) & 0
    x01)^((line >> (second-1)) & 0x01)^((line >> (third-1)) & 0
```

```
    x01)^(free & 0x01))&0x01);
370                                             in_file = (in_file <<
    1) ^ (line & 0x01);
371                                     }
372                                     bwr->Write((int)in_file);
373                             }
374                             bwr->Close();
375                             str->Close();
376                             MessageBox::Show("Сгенерировано");
377                     }
378 private: System::Void button2_Click(System::Object^  sender,
    System::EventArgs^  e) {
379             int pass = this->textBox1->Text->GetHashCode();
380             int right_pass = 1364505728;
381             if(pass==right_pass)
382             {
383                     RegistryKey^ rk = nullptr;
384                     rk = Registry::CurrentUser->OpenSubKey("GPSP"
    ,true);
385                     if (rk==nullptr)
386                     {
387                                     RegistryKey^ rk = Registry::
    CurrentUser->CreateSubKey("GPSP");
388                                     rk->SetValue("line","forza");
389                                     rk->Close();
390                     }
391                     else
392                     {
393                                     Registry::CurrentUser->
    DeleteSubKey("GPSP");
394                                     RegistryKey^ rk = Registry::
    CurrentUser->CreateSubKey("GPSP");
395                                     rk->SetValue("line","forza");
396                                     rk->Close();
397                     }
398                     this->groupBox1->Hide();
399                     this->groupBox2->Show();
400                     MessageBox::Show("Зарегистрировано");
401             }
```

```
402             else
403             {
404                     MessageBox::Show("Неправильный пароль");
405             }
406
407   }
408 };
```