МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**Вятский государственный университет**»

**(«ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №7

по дисциплине «Технологии программирования»

Выполнил студент группы ИВТ-21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Щесняк Д. С./

Проверил доцент кафедры ЭВМ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М. Л./

Киров 2016

1. Задание на лабораторную работу

Необходимо разработать многопоточное приложение решающее проблему производителя-потребителя с использованием циклического буфера. В данной задаче есть 2 производителя и потребитель. Производители «кладут» данные в циклический буфер, а потребитель достает из него.

1. Словесное описание алгоритма работы

Запускается основной поток, который порождает 3 дочерних потока. Два потока для двух производителей и один поток для потребителя. Буферная структура имеет указатель для ввода данных и для считывания данных. При вводе данных потоком в буфер указатель ввода сдвигается. При считывании данных потребителем, указатель считывания сдвигается. Буфер является пустым если равны указатели записи и считывания. Разграничение к записи в буфер между двумя производителями производится при помощи объекта Mutex.

1. Экранные формы

Экранные формы представлены на рисунках 1, 2.

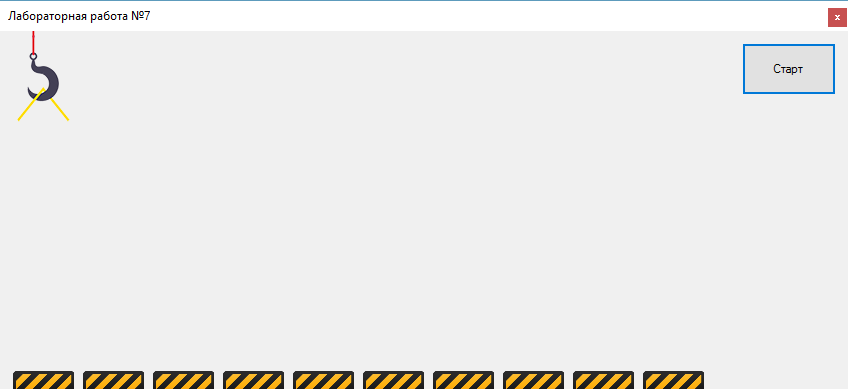


Рисунок 1 – Окно программы в начальный момент времени

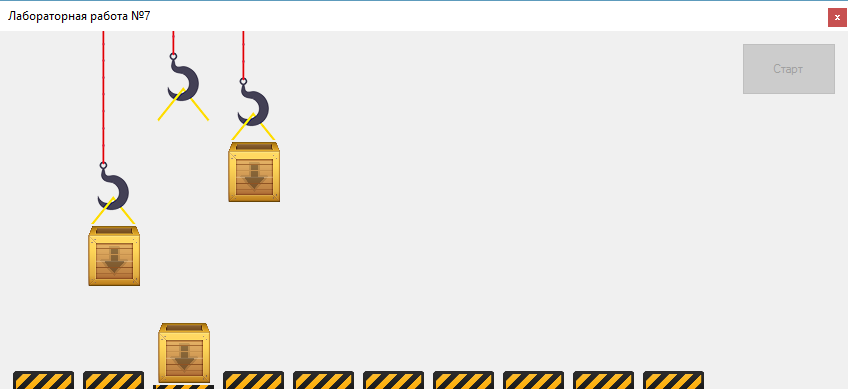


Рисунок 2 – Окно программы во время ее выполнения

1. Исходный код программы

Исходный код программы представлен на рисунке 3. Исходный код классов и вспомогательных функций представлен на рисунке 4

|  |
| --- |
| #pragma once  #include <windows.h>  #include <process.h>  #include "classes.h"  namespace AnimateLab7 {    const int LEFT = 12;  const int HOOK\_TOP = -266;  using namespace System;  using namespace System::ComponentModel;  using namespace System::Collections;  using namespace System::Windows::Forms;  using namespace System::Data;  using namespace System::Drawing;  using namespace System::Threading;  /// <summary>  /// Сводка для MainForm  /// </summary>  public ref class MainForm : public System::Windows::Forms::Form  {  public:  MainForm(void)  {  SetStyle(ControlStyles::SupportsTransparentBackColor, true);  SetStyle(ControlStyles::DoubleBuffer, true);  SetStyle(ControlStyles::UserPaint, true);  SetStyle(ControlStyles::AllPaintingInWmPaint, true);  InitializeComponent();  System::Windows::Forms::Control::CheckForIllegalCrossThreadCalls = false;  //  }  protected:  /// <summary>  /// Освободить все используемые ресурсы.  /// </summary>  ~MainForm()  {  if (components)  {  delete components;  }  }  private: System::Windows::Forms::PictureBox^ pictureBox1;  private: System::Windows::Forms::PictureBox^ pictureBox2;  private: System::Windows::Forms::PictureBox^ pictureBox3;  private: System::Windows::Forms::PictureBox^ pictureBox4;  private: System::Windows::Forms::PictureBox^ pictureBox5;  private: System::Windows::Forms::PictureBox^ pictureBox6;  private: System::Windows::Forms::PictureBox^ hookProd1;  private: System::Windows::Forms::PictureBox^ Box0;  private: System::Windows::Forms::PictureBox^ hookProd2;  private: System::Windows::Forms::PictureBox^ hookCons;  private: System::Windows::Forms::Button^ startBut;  private: System::Windows::Forms::PictureBox^ Box1;  private: System::Windows::Forms::PictureBox^ Box2;  private: System::Windows::Forms::PictureBox^ Box3;  private: CirclBuffer^ buf;  protected:  private:  /// <summary>  /// Обязательная переменная конструктора.  /// </summary>  System::ComponentModel::Container ^components;  #pragma region Windows Form Designer generated code  /// <summary>  /// Требуемый метод для поддержки конструктора — не изменяйте  /// содержимое этого метода с помощью редактора кода.  /// </summary>  void InitializeComponent(void)  {  System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MainForm::typeid));  this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());  this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());  this->hookProd1 = (gcnew System::Windows::Forms::PictureBox());  this->Box0 = (gcnew System::Windows::Forms::PictureBox());  this->hookProd2 = (gcnew System::Windows::Forms::PictureBox());  this->hookCons = (gcnew System::Windows::Forms::PictureBox());  this->startBut = (gcnew System::Windows::Forms::Button());  this->Box1 = (gcnew System::Windows::Forms::PictureBox());  this->Box2 = (gcnew System::Windows::Forms::PictureBox());  buf = gcnew CirclBuffer();  }  #pragma endregion  private: System::Void startBut\_Click(System::Object^ sender, System::EventArgs^ e) {  Producer^ prod1 = gcnew Producer(hookProd1, buf);  Producer^ prod2 = gcnew Producer(hookProd2, buf);  Consumer^ cons = gcnew Consumer(hookCons, buf);  Thread^ t = gcnew Thread(gcnew ThreadStart(prod1, &Producer::run));  t->Start();  Thread^ t2 = gcnew Thread(gcnew ThreadStart(cons, &Consumer::run));  t2->Start();  Thread^ t3 = gcnew Thread(gcnew ThreadStart(prod2, &Producer::run));  t3->Start();  startBut->Enabled = false;  }  };  } |

Рисунок 3 – Исходный код программы

|  |
| --- |
| #pragma once  #include <windows.h>  #include <math.h>  ref class CirclBuffer  {  public:  CirclBuffer() {  addMut = CreateMutex(NULL, false, NULL);  data = gcnew array<System::Windows::Forms::PictureBox^>(10);  in = 0;  out = 0;  };  ~CirclBuffer() {};  bool isFull() {  return (in + 1) % 10 == out;  }  bool isEmpty() {  return in == out;  }  int getPlace() {  WaitForSingleObject(addMut, INFINITE);  return in;  }  int add(System::Windows::Forms::PictureBox^ pic) {  data[in] = pic;  int res = in;  in = (in + 1) % 10;  ReleaseMutex(addMut);  return res;  }  System::Windows::Forms::PictureBox^ get(int &pos) {  pos = out;  out = (out + 1) % 10;  return data[pos];  }  private:  array<System::Windows::Forms::PictureBox^>^ data;  int in;  int out;  HANDLE addMut;  };  ref class Producer  {  public:  Producer(System::Windows::Forms::PictureBox^ picture, CirclBuffer^ buffer){  hook = picture;  hSync = CreateMutexW(NULL, false, NULL);  buf = buffer;  speed = 1;  };  ~Producer() {};  void run() {  for (int i = 0; i < 50; i++)  {  int pos = buf->getPlace();  while (buf->isFull()) {};  System::Windows::Forms::PictureBox ^ pic = (System::Windows::Forms::PictureBox^)hook->FindForm()->Controls->Find("Box" + (pos % 10), true)[0];  this->moveX((pos % 10));  this->putBox(pic);  buf->add(pic);  }  }  void moveX(int pos) {  pos = 70 \* pos + 12;  int dx = 1;  if (hook->Location.X != pos) {  dx = (pos - hook->Location.X) / abs(pos - hook->Location.X);  }  while (hook->Location.X != pos) {  hook->Location = System::Drawing::Point(hook->Location.X + dx, hook->Location.Y);  Sleep(speed);  }  }  void putBox(System::Windows::Forms::PictureBox^ box) {  box->Location = System::Drawing::Point(hook->Location.X, -100);  //box->Visible = true;  while (hook->Location.Y != -450) {  hook->Location = System::Drawing::Point(hook->Location.X, hook->Location.Y - 1);  //box->Location = System::Drawing::Point(box->Location.X, box->Location.Y - 1);  Sleep(speed);  }  while (hook->Location.Y != -60) {  hook->Location = System::Drawing::Point(hook->Location.X, hook->Location.Y + 1);  box->Location = System::Drawing::Point(box->Location.X, box->Location.Y + 1);  Sleep(speed);  }  while (hook->Location.Y != -266) {  hook->Location = System::Drawing::Point(hook->Location.X, hook->Location.Y - 1);  Sleep(speed);  }  }  private:  System::Windows::Forms::PictureBox^ hook;  CirclBuffer^ buf;  HANDLE hSync;  int speed;  };  ref class Consumer  {  public:  Consumer(System::Windows::Forms::PictureBox^ picture, CirclBuffer^ buffer) {  hook = picture;  buf = buffer;  speed = 15;  };  ~Consumer() {};  void run() {  for (int i = 0; i < 50; i++)  {  while (buf->isEmpty()) {};  int pos;  System::Windows::Forms::PictureBox^ pic = buf->get(pos);  this->moveX(pos);  this->takeBox(pic);  }  }  void moveX(int pos) {  pos = 70 \* pos + 12;  int dx = 1;  if (hook->Location.X != pos) {  dx = (pos - hook->Location.X) / abs(pos - hook->Location.X);  }  while (hook->Location.X != pos) {  hook->Location = System::Drawing::Point(hook->Location.X + dx, hook->Location.Y);  Sleep(speed);  }  }  void takeBox(System::Windows::Forms::PictureBox^ box) {  while (hook->Location.Y != -60) {  hook->Location = System::Drawing::Point(hook->Location.X, hook->Location.Y + 1);  Sleep(speed);  }  while (hook->Location.Y != -450) {  hook->Location = System::Drawing::Point(hook->Location.X, hook->Location.Y - 1);  box->Location = System::Drawing::Point(box->Location.X, box->Location.Y - 1);  Sleep(speed);  }  while (hook->Location.Y != -266) {  hook->Location = System::Drawing::Point(hook->Location.X, hook->Location.Y + 1);  Sleep(speed);  }  }  private:  System::Windows::Forms::PictureBox^ hook;  CirclBuffer^ buf;  int speed;  }; |

Рисунок 4 – Классы и вспомогательные функции

1. Вывод

В ходе выполнения лабораторной работы были полученные необходимые знания базового синтаксиса языка программирования С++. Было изучено объектно-ориентированное программирование и многопоточное программирование. На основе этих знаний была решена задача синхронизации трех потоков в задаче «поставщики-потребители» Данные знания являются фундаментальными и необходимы для дальнейшего продолжения изучения различных технологий программирования и языка С++.