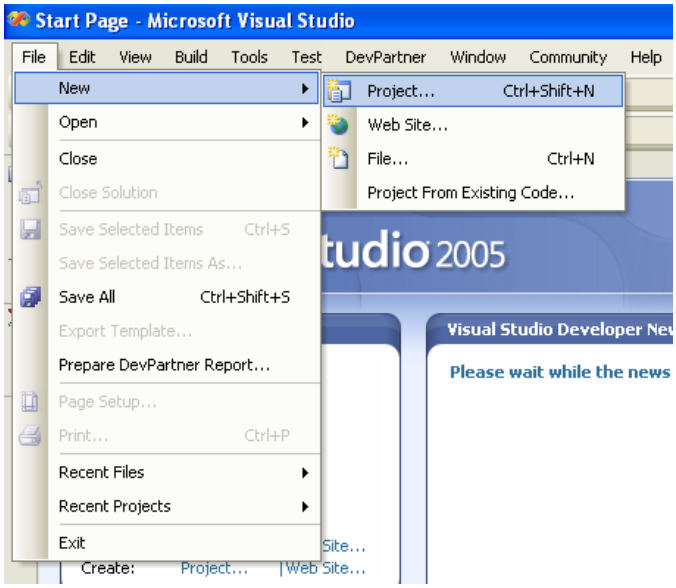
Лабораторная работа №2

Базовые алгоритмы: очередь и стек.

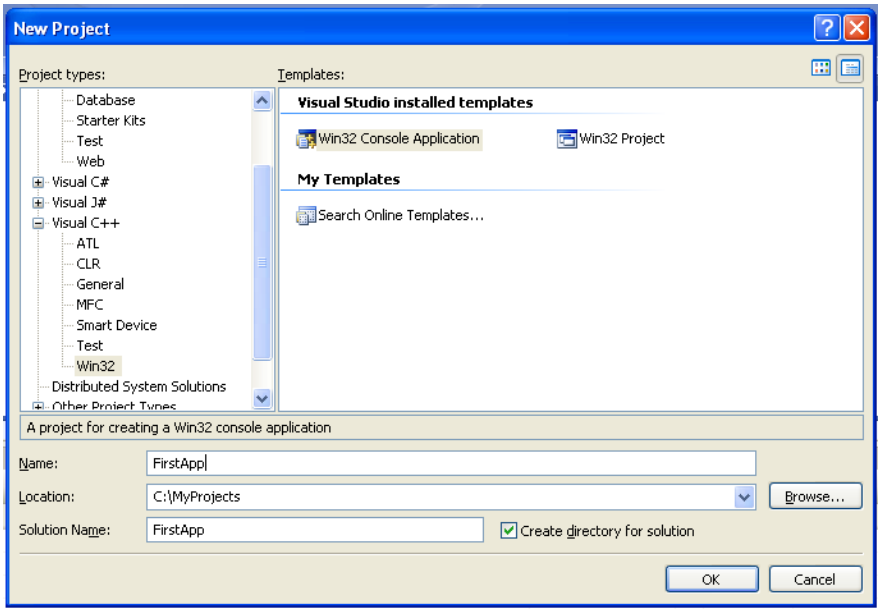
Цель работы: освоить алгоритмы работы очереди и стека.

Ход работы:

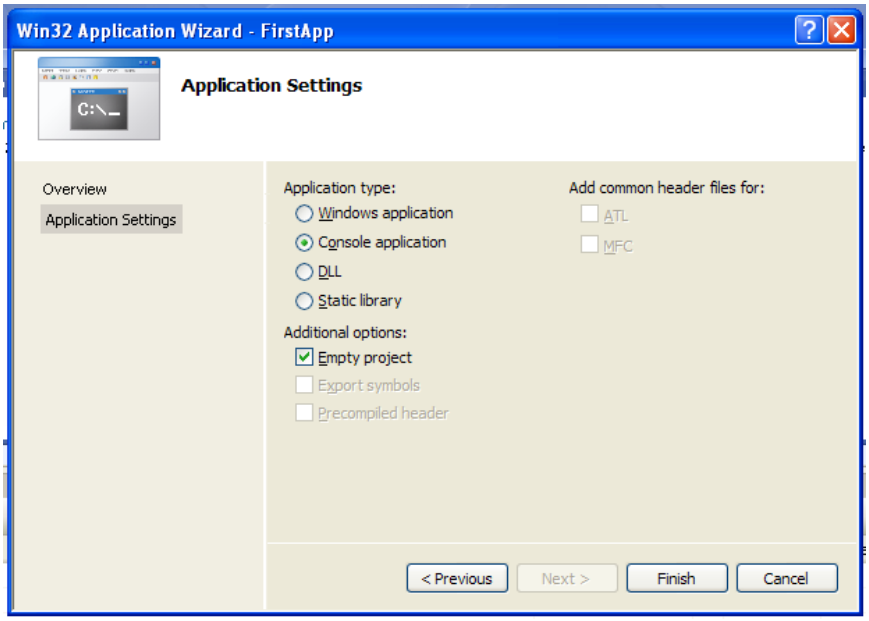
1. Создаем новый проект.



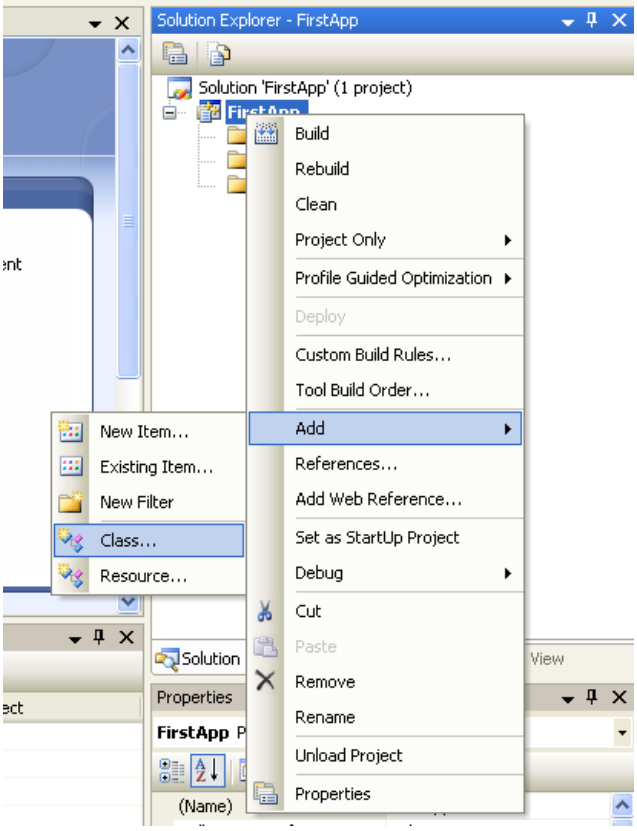
2. Выбираем тип проекта «Win32», консольное приложение «Win32 Console Application». Назначаем имя проекту.

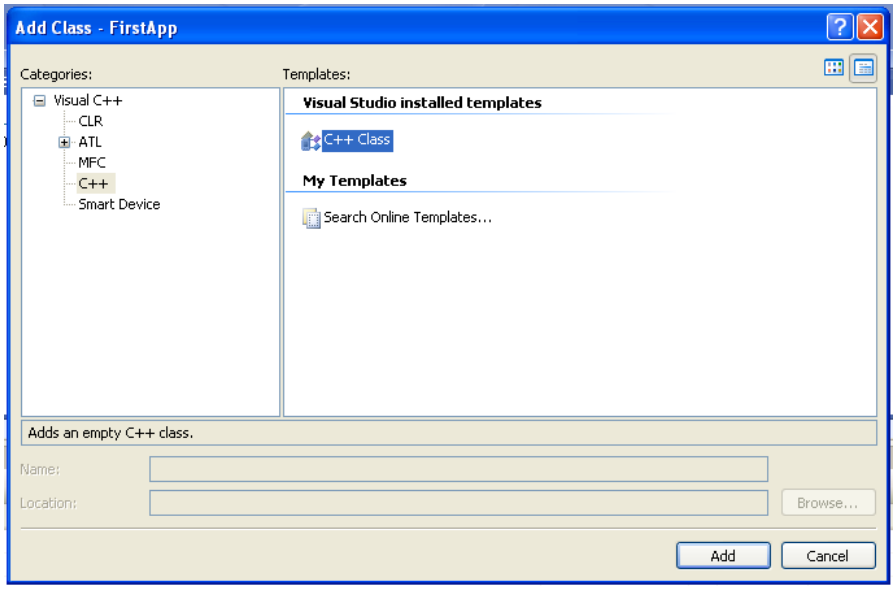


3. На вкладке «Application Settings» выбираем тип «Console Application». Проект должен быть пустым. За это отвечает галочка «Empty project».

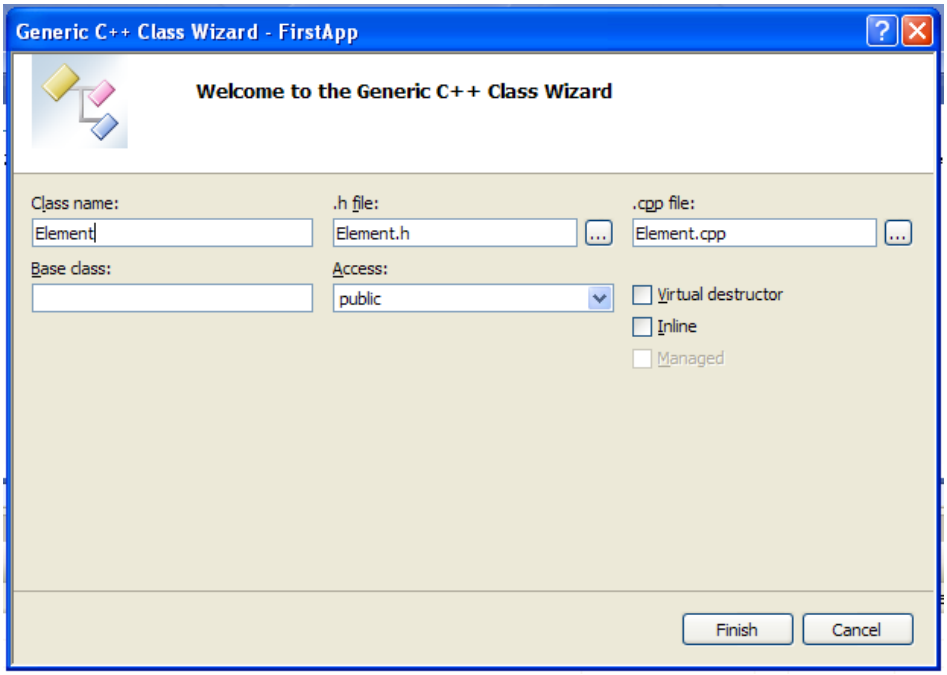


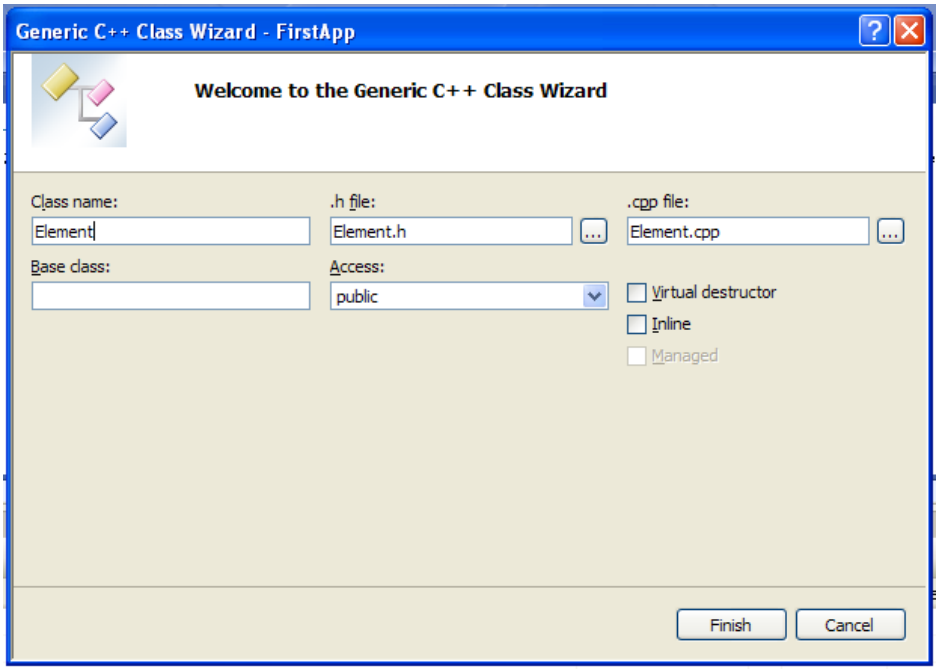
4. В окне «Solution Explorer» добавляем к проекту новый класс.



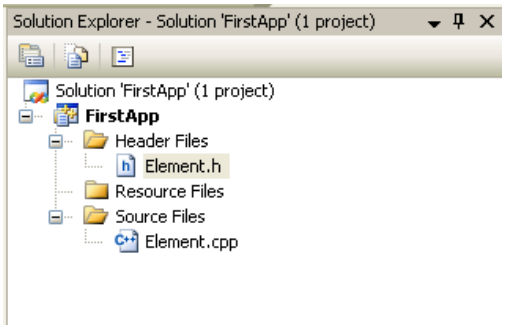


5. Даем классу имя Element. Необходимые имена файлов заполнятся автоматически.





6. Файлы Element.h и Element.cpp будут добавлены в проект.



7. Удалите из проекта файл Element.cpp, так как класс Element будет содержать только данные.

Создайте в классе необходимые данные для реализации класса Element (файл Element.h):

```
#pragma once

class Element
{
public:
  Element* prev;
  Element* next;
  int value;

  Element() { prev = NULL; next = NULL; value = 0; };
};
```

8. Создаем новый класс, отвечающий за очередь. Даем ему имя Queue. Описание класса будет размещаться в файле Queue.h, реализация – в файле Queue.cpp.
9. Добавляем в описание методы Put, Get и Print для помещения элемента в очередь, удаления из очереди и распечатки содержимого очереди. Файл Queue.h должен выглядеть так:

```
#pragma once
```

```
#include "Element.h"

class Queue
{
public:
  Element* first;
  Element* last;
  int count;

  Queue(void);
  ~Queue(void);
  void Put(Element* e);
  Element* Get();
  void Print();
};
```

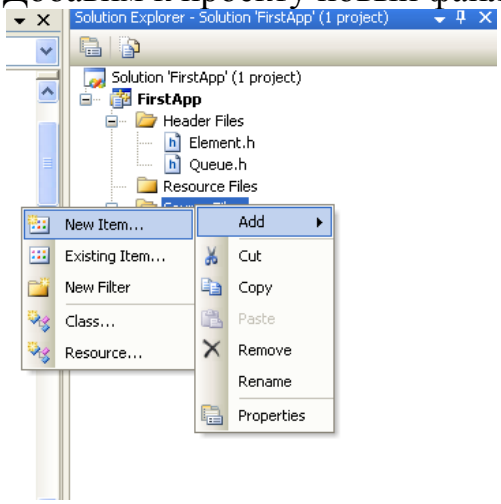
10. Обратите внимание на строку #include “Element.h”. Описание класса Element подключается, т.к. в классе Queue используются переменные класса Element.

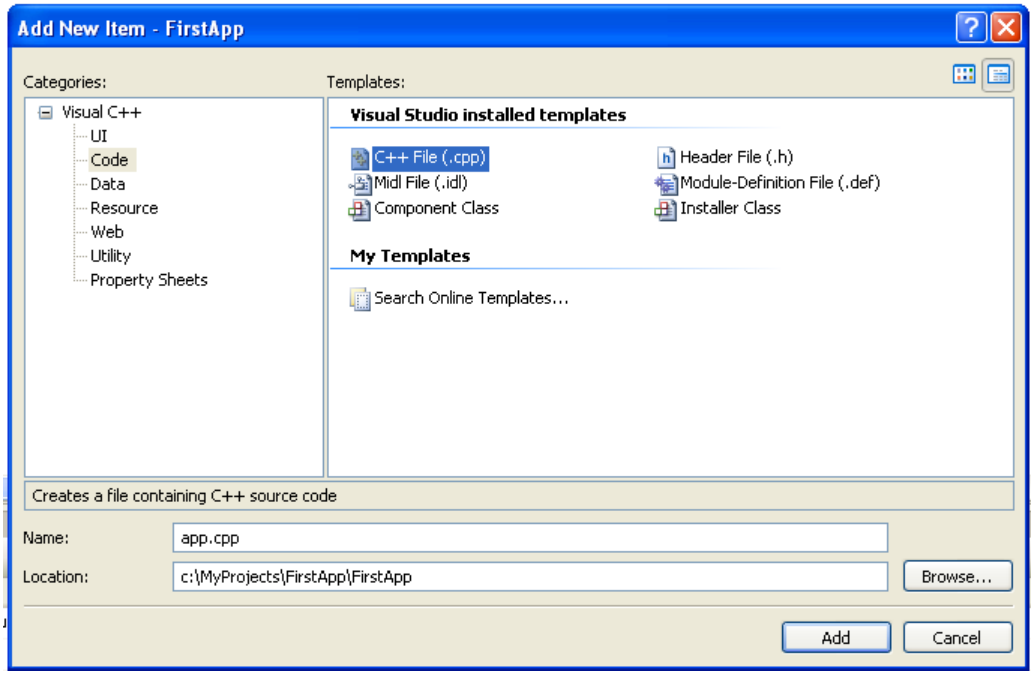
Реализуем методы класса в файле Queue.cpp:

```
#include "Queue.h"
#include <iostream>

using namespace std;

Queue::Queue(void)
{
  first = 0;
  last = 0;
  count = 0;
}

Queue::~Queue(void)
{
}

void Queue::Put(Element* e)
{
  if (e == 0)
    return;

  e->prev = last;

  if (count < 1)
  {
    last = e;
    first = e;
  }

  last->next = e;
  last = e;
  count++;

  cout << "Element " << e->value << "added to queue" << endl;
}

Element* Queue::Get()
{
  Element* e;

  if (first == 0)
    return 0;
```

```
  e = first;
  first = first->next;
  first->prev = 0;
  e->next = 0;
  count--;

  cout << "Element " << e->value << "removed from queue" << endl;
  return e;
}

void Queue::Print()
{
  if(count < 1)
  {
    cout << "Queue is empty";
    return;
  }

  Element* e;
  e = first;
  while(e != 0)
  {
    cout << e->value << ", ";
    e = e->next;
  }
  cout << endl;
}
```

11. Обратите внимание, что при использовании cout необходимо подключить библиотеку <iostream> и объявить использование пространства имен std (using namespace std;).
12. Добавим к проекту новый файл:





Присвоим ему имя app.cpp. В этом файле будет располагаться основная программа.

13. Создадим 5 элементов и добавим их в очередь в порядке: e2, e3, e1, e5, e4. Далее распечатываем очередь. После чего забираем 2 элемента из очереди и распечатываем очередь снова.

```
#include "Element.h"
#include "Queue.h"

#include <conio.h>

void main ()
{
  Element e1, e2, e3, e4, e5;
  e1.value = 10;
  e2.value = 20;
  e3.value = 30;
  e4.value = 40;
  e5.value = 50;

  Queue q;

  q.Put(&e2);
  q.Put(&e3);
  q.Put(&e1);
  q.Put(&e5);
  q.Put(&e4);

  q.Print();

  q.Get();
  q.Get();

  q.Print();

  getch();}
```

14. Обратите внимание на то, что к программе подключены описания классов Element и Queue, т.к. они используются.

14. Запустите программу на выполнение. Убедитесь, что она ведет себя правильно.
15. Создайте новый проект, в котором будет реализован класс стека (Stack) вместо класса очереди. Создайте основную программу и покажите, что класс работает.
16. Выполните индивидуальное задание.
17. Оформите отчет по работе, описав принцип работы алгоритмов очереди и стека, поместите в отчет блок-схему реализации метода из индивидуального задания, ответьте в отчете на контрольные вопросы.

Индивидуальные задания:

1. Реализуйте в классе Stack метод, который выводит на экран сумму всех элементов стека.
2. Реализуйте в классе Queue метод, который выводит на экран сумму всех элементов.
3. Реализуйте в классе Stack метод, который выводит на экран произведение всех элементов.
4. Реализуйте в классе Queue метод, который выводит на экран произведение всех элементов.
5. Реализуйте в классе Stack метод, который выводит сумму первых трех элементов стека. Если элементов меньше, то сумму всех элементов.
6. Реализуйте в классе Queue метод, который выводит сумму первых трех элементов очереди. Если элементов меньше, то сумму всех элементов.
7. Реализуйте в классе Stack метод, который выводит произведение первых трех элементов стека. Если элементов меньше, то выводит сообщение о невозможности расчета.
8. Реализуйте в классе Queue метод, который выводит произведение первых трех элементов очереди. Если элементов меньше, то выводит сообщение о невозможности расчета.
9. Реализуйте в классе Stack метод, который выводит сумму всех нечетных элементов стека. Если элементов меньше, то сумму всех элементов.
10. Реализуйте в классе Queue метод, который выводит сумму всех четных элементов очереди. Если элементов меньше, то выводит сообщение о невозможности расчета.

Контрольные вопросы:

1. Нарисуйте блок-схему реализации метода Queue::Get. Объясните работу метода.
2. Нарисуйте блок-схему реализации метода Queue::Put. Объясните работу метода.
3. Нарисуйте блок-схему реализации метода Queue::Print. Объясните работу метода.

4. Нарисуйте блок-схему реализации метода Stack::Get. Объясните работу метода.
5. Нарисуйте блок-схему реализации метода Stack::Put. Объясните работу метода.
6. Нарисуйте блок-схему реализации метода Stack::Print. Объясните работу метода.