Лабораторная работа № 1

Формирование массивов экспериментальных данных

Цель работы: освоить принципы формирования монотонных (упорядоченных и упорядоченных в обратном порядке), частично упорядоченных и случайных последовательностей данных; изучить функции, позволяющие производить оценку длительности времени выполнения алгоритмов.

Общие сведения

Изучение и исследование разнообразных алгоритмов сортировки и поиска, структур данных, невозможно без использования либо *реальных* массивов данных, либо формируемые *искусственно*, но обладающих свойствами, схожими со свойствами реальных массивов данных. Так как у исследователя не всегда имеется возможность использования реальных данных, то возникает естественная необходимость в формировании псевдо-реальных массивов данных, обладающих теми или иными свойствами.

При исследовании алгоритмов сортировки, как правило, оговаривается характер упорядочивания – по возрастанию или убыванию. При этом под монотонной последовательностью (массивом) данных понимается последовательность, для *всех* элементов которой выполняется отношение $D_i \le D_{i+1}$ (либо $D_i \ge D_{i+1}$). Если для всех элементов выполняется отношение $D_i < D_{i+1}$ (либо $D_i > D_{i+1}$), то последовательность называется строго монотонной. Если последовательность данных является монотонной и характер отношений ее элементов соответствует цели алгоритма сортировки, то последовательность называется упорядоченной, в противном случае – упорядоченной в обратном порядке.

Случайной последовательностью называется последовательность, элементы которой являются случайными величинами (как правило, псевдослучайными), распределенными по некоторому закону распределения. Наиболее часто для формирования псевдослучайных величин используется равномерный закон, но на его основе можно сформировать псевдослучайные величины, распределенные и по другим законам распределения – экспоненциальному, нормальному и др.

Помимо упорядоченных и случайных последовательностей существуют также последовательности, элементы которых обладают свойством упорядоченности в каком-либо смысле. Для определенности, будем называть такие последовательности частично упорядоченными. Можно выделить следующие, наиболее характерные случаи частично упорядоченных последовательностей: «пилообразные», «синусоидальные», «ступенчатые», «квази-упорядоченные» (см. приложение А). В первом случае всю последовательность можно разбить на интервалы (длины интервалов, для простоты, можно считать одинаковыми), в пределах которых отношение порядка выполняется, но оно может быть не справедливо для соседних элементов, принадлежащих двум соседним интервалам. Во втором случае отношение порядка на соседних интервалах чередуется, например, для k-го интервала справедливо $D_i \le D_{i+1}$, для k+1-го – $D_i \ge D_{i+1}$. В третьем случае в некотором смысле упорядочены сами интервалы – максимальное значение на k-м интервале меньше (не больше), чем минимальное значение на k+1-м интервале. Квази-

упорядоченная последовательность – это последовательность, в которой имеется ограниченное количество инверсий (количество элементов, для которых не выполняется отношение порядка) и оно много меньше размера последовательности.

В качестве исходных последовательностей для исследования алгоритмов сортировки и поиска наиболее часто используют последовательности целых чисел и чисел с плавающей запятой. В этом случае для формирования псевдослучайной величины можно использовать функции стандартной библиотеки[1]:

`int rand(void);` – возвращает псевдослучайную величину в диапазоне от `0` до `RAND_MAX`;

`int random(int N);` – возвращает псевдослучайную величину в диапазоне от `0` до `N-1`.

Используя указанные функции можно получить и псевдослучайные числа с плавающей запятой, принадлежащие некоторому заданному диапазону.

Оценку временных характеристик алгоритмов сортировки и поиска можно производить с использованием функции:

```
time_t time(time_t *timer);
```

Однако данная функция позволяет оценивать временные интервалы с точностью до секунды. Поэтому в ряде случаев более приемлемо использование следующей функции прикладного программного интерфейса Windows (для этого необходимо подключить заголовочный файл `windows.h`):

```
void GetLocalTime(SYSTEMTIME * st);
```

В качестве параметра эта функция получает указатель на структуру `SYSTEMTIME`, поля которой содержат текущие значения (на момент вызова функции) миллисекунд, секунд, минут, …, года, что позволяет оценивать временные характеристики с точностью до миллисекунд.

Еще одной функцией, которая позволяет оценить время выполнения, является функция

```
DWORD GetTickCount(void);
```

Эта функция возвращает количество «тиков» (миллисекунд), прошедших с момента запуска операционной системы.

Выполнение лабораторных работ можно осуществлять в среде Visual Studio. Краткая инструкция, позволяющая создать пустой начальный проект представлена в приложении Б.

Задание

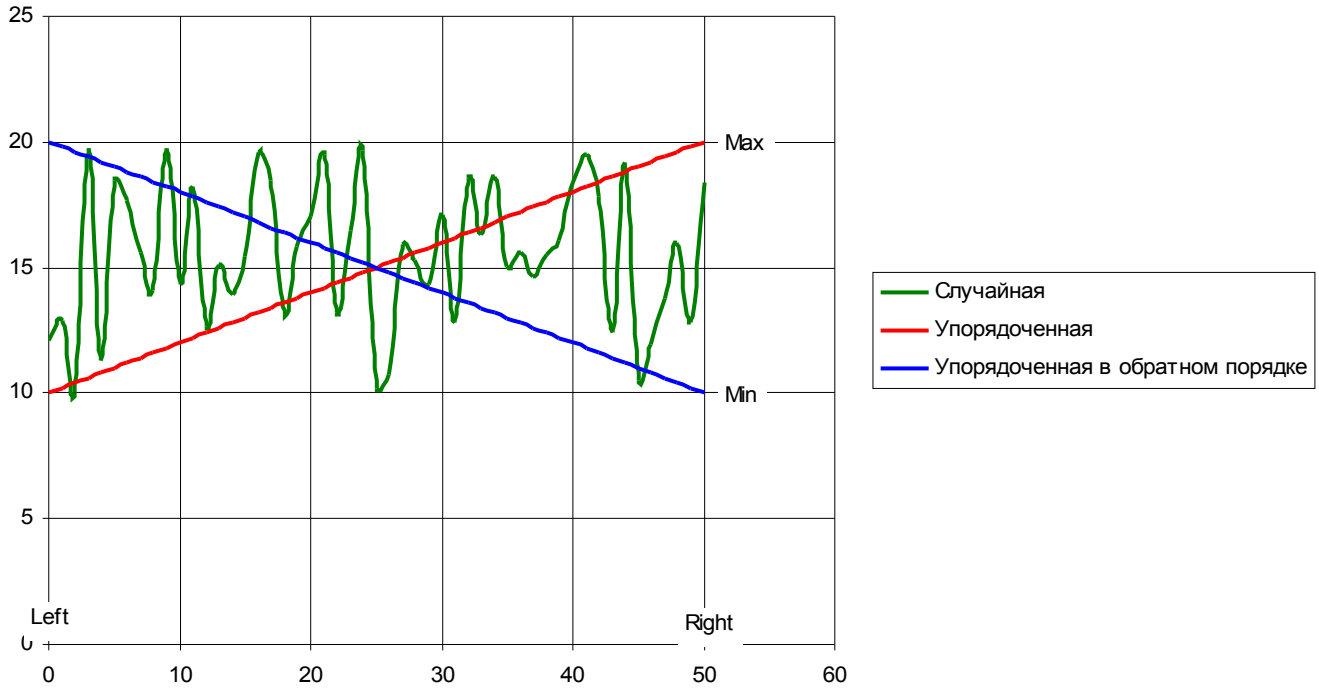
1. Разработать функции, формирующие упорядоченные (в том числе и в обратном порядке), частично упорядоченные и случайные последовательности целых чисел и чисел с плавающей запятой. При реализации функций считать, что выделение памяти под последовательности происходит вне этих функций, а в качестве формальных параметров функции получают указатель на массив, его размер и, возможно, диапазон изменения величин и длину интервалов (для частично упорядоченных последовательностей). Рекомендуется продумать единый прототип для функций, формирующих последовательности, определить

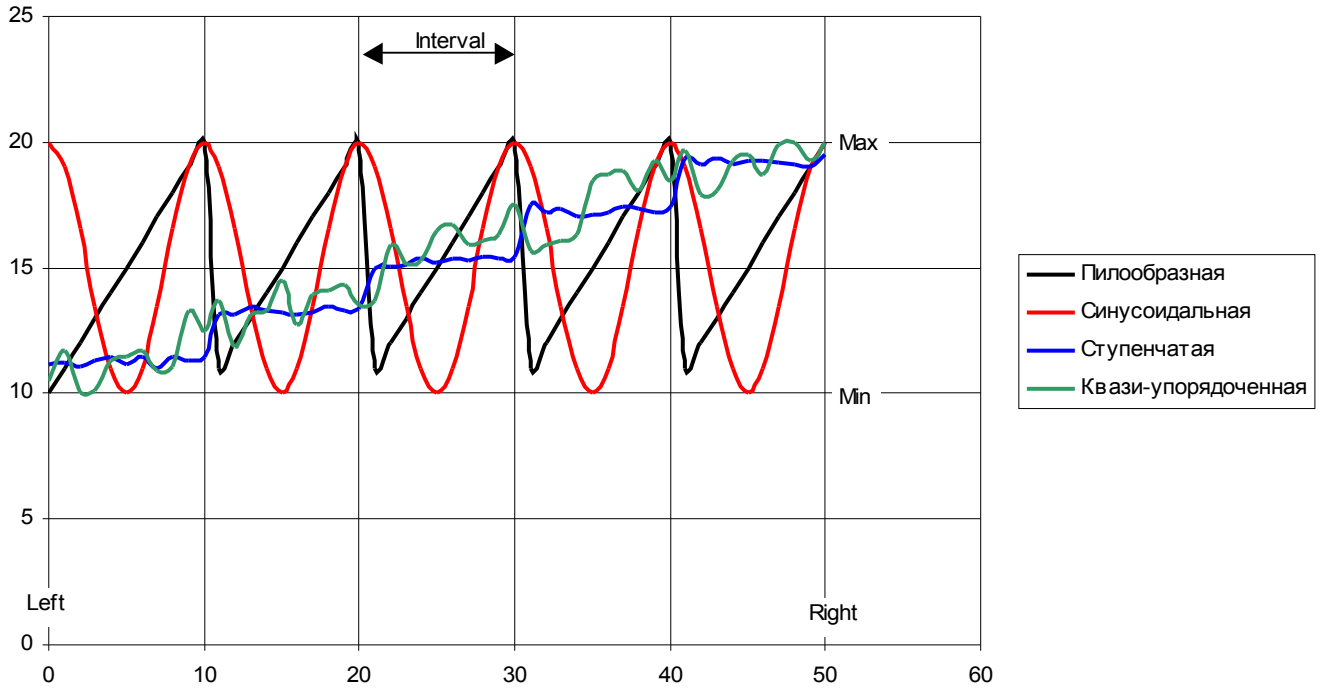
[1] Предполается, что реализация алгоритмов будет осуществляться на языке C (C++)

его и использовать массив указателей на функции для автоматизации сбора статистической информации.

2. Оценить длительность формирования последовательностей всех типов для нескольких значений размеров последовательностей ($5 \cdot 10^5$, $10 \cdot 10^5$, …, $50 \cdot 10^5$), и на основе полученных значений построить графики зависимостей длительностей формирования от размера последовательностей.
3. Составить отчет, в котором привести блок-схемы функций (возможно, на примере одной функции), реализующих формирование частично упорядоченных последовательностей, и главной функции, полученные графики зависимостей и выводы по полученным результатам.

Приложение А. Виды последовательностей





Приложение Б. Создание проекта в Visual Studio

Процесс создания консольного приложения в среде разработки Visual Studio представлен на следующих рисунках.

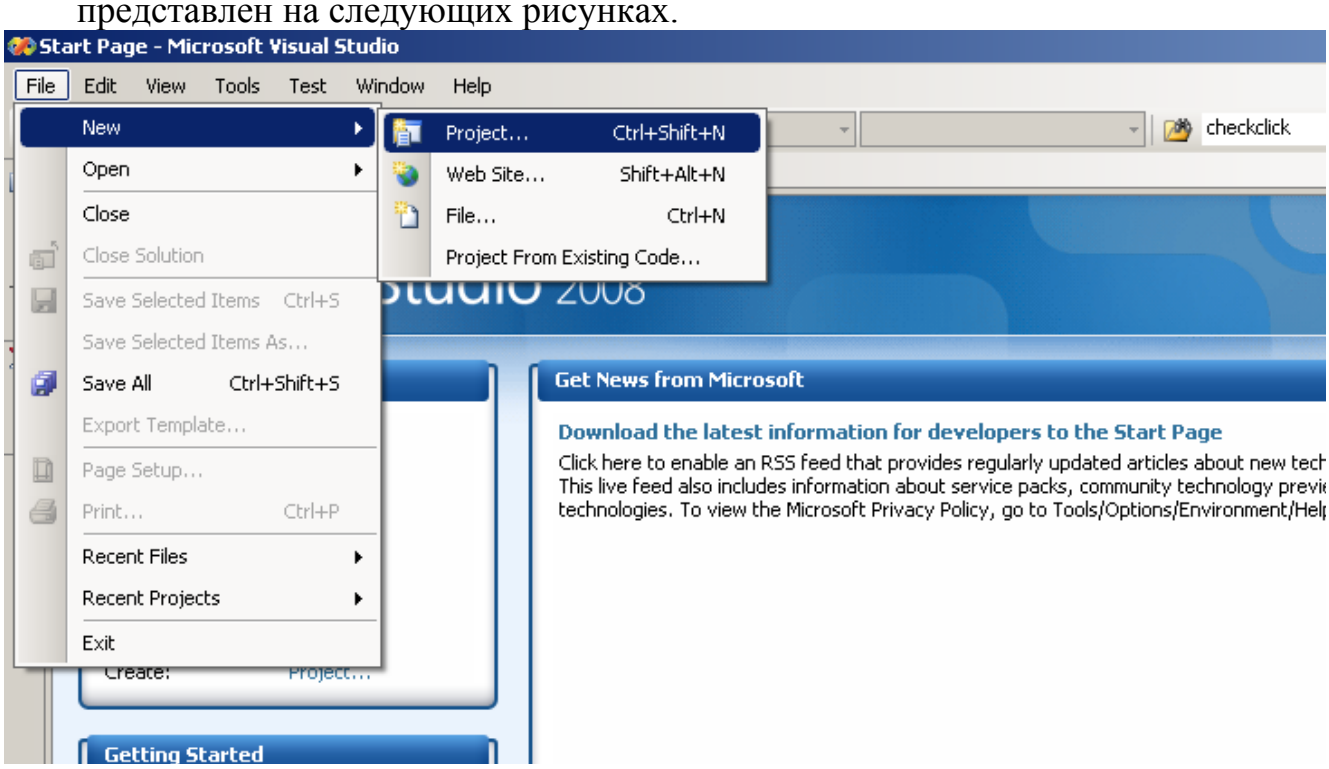


Рис. Б. 1. Выбор типа создаваемого объекта – проект

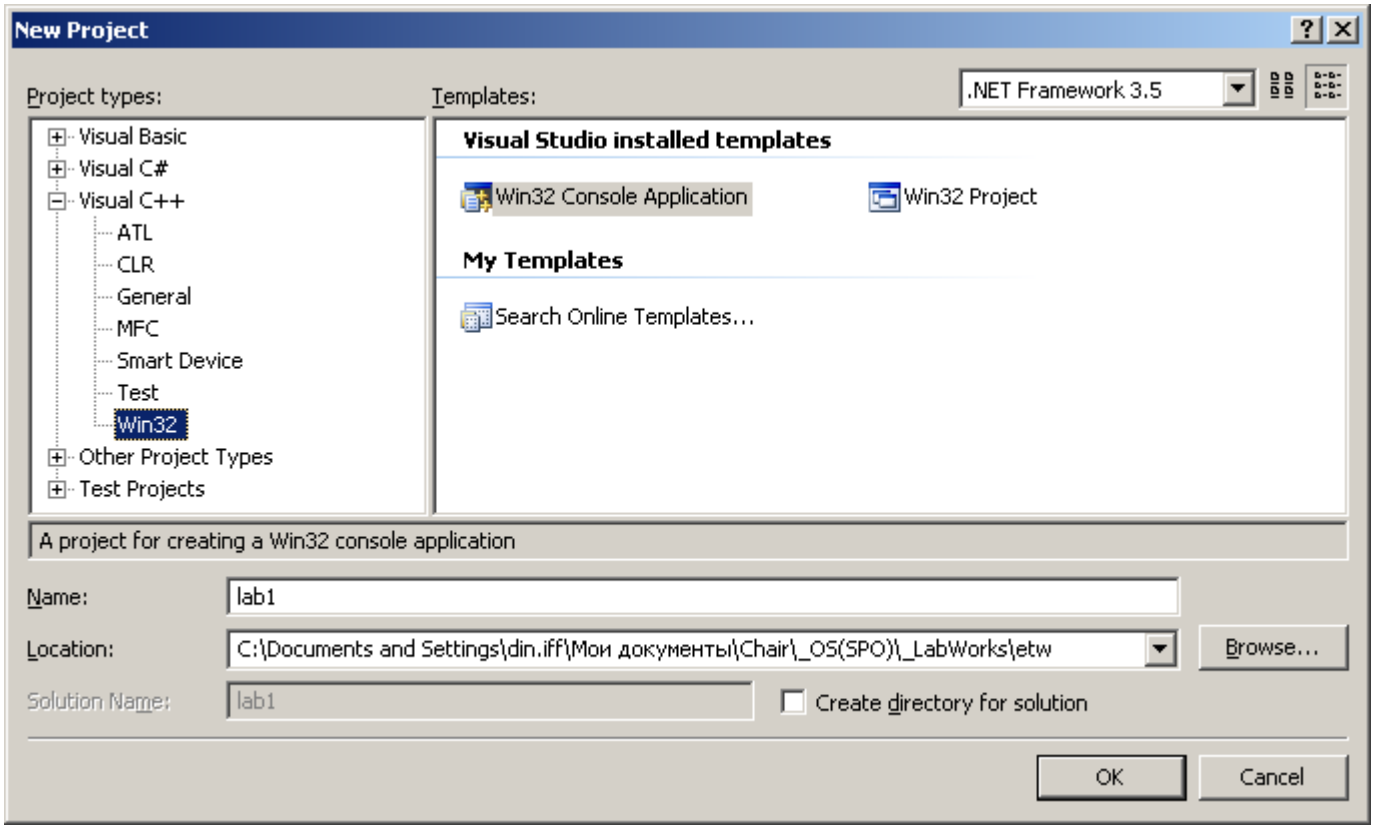


Рис. Б. 2. Выбор типа приложения – Win32, Win32 Console Application, а также указание *имени проекта* и его месторасположения

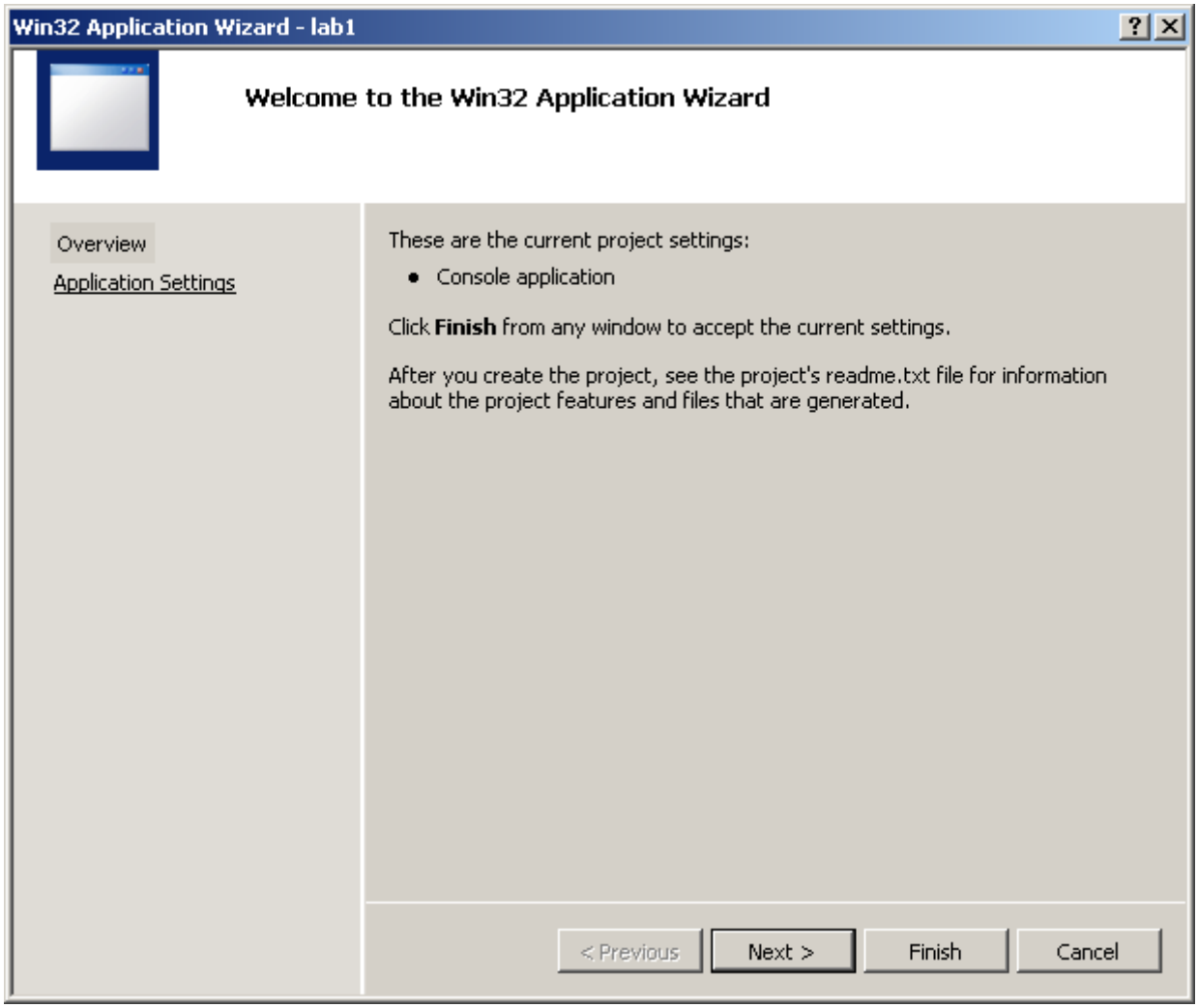


Рис. Б. 3. Выбор закладки настроек приложения (Application Settings)

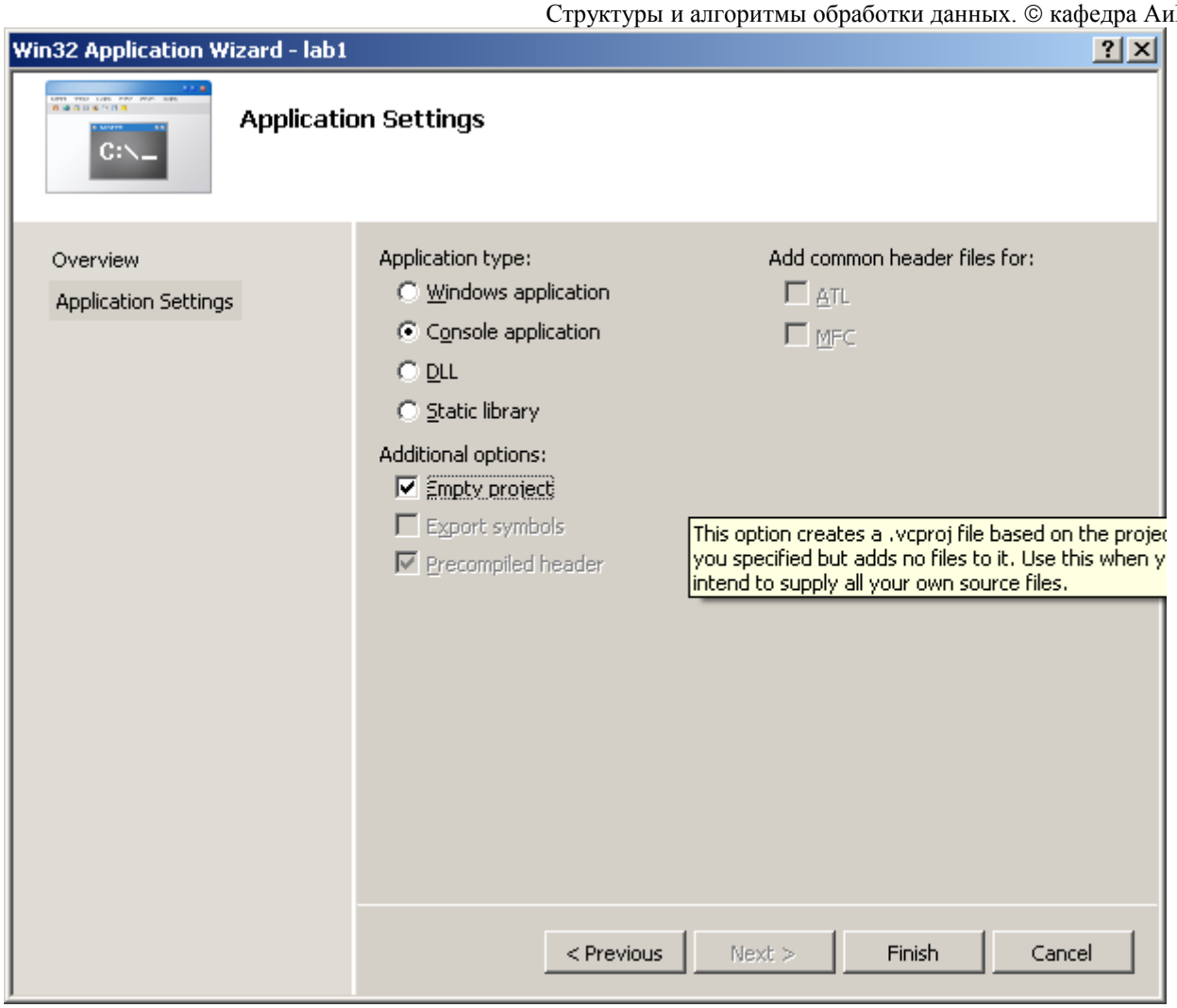


Рис. Б. 4. Задание настроек – Console Application и Empty Project

После этого необходимо создать как минимум один модуль (unit). Для этого в обозревателе решения (Solution Explorer), необходимо правой кнопкой мыши щелкнуть на ветви Source Files (исходные файлы) и выбрать необходимый пункт в контекстном меню (рис. Б. 5 и рис. Б. 6)

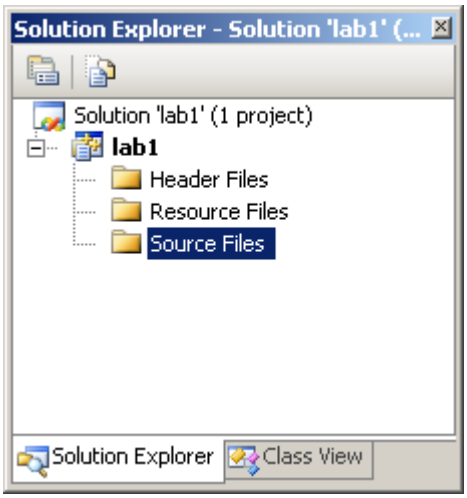


Рис. Б. 5. Выбор типа добавляемого файла в обозревателе решения

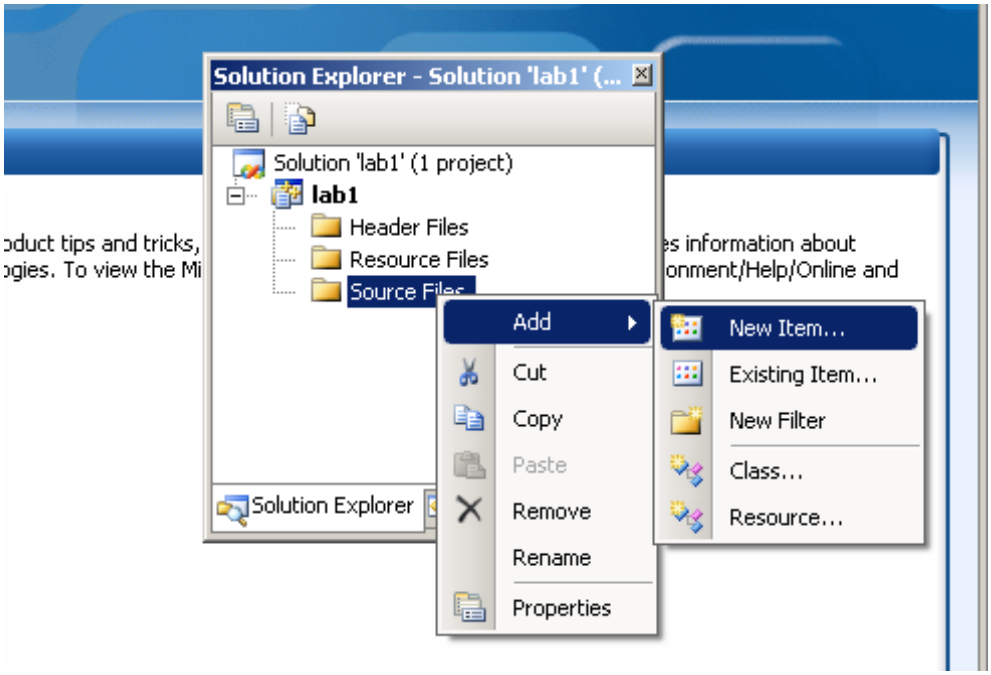


Рис. Б. 6. Выбор типа операции и категории добавляемого объекта

В появившемся диалоговом окне необходимо указать тип файла (C++) и задать имя создаваемого модуля (рис. Б. 7).

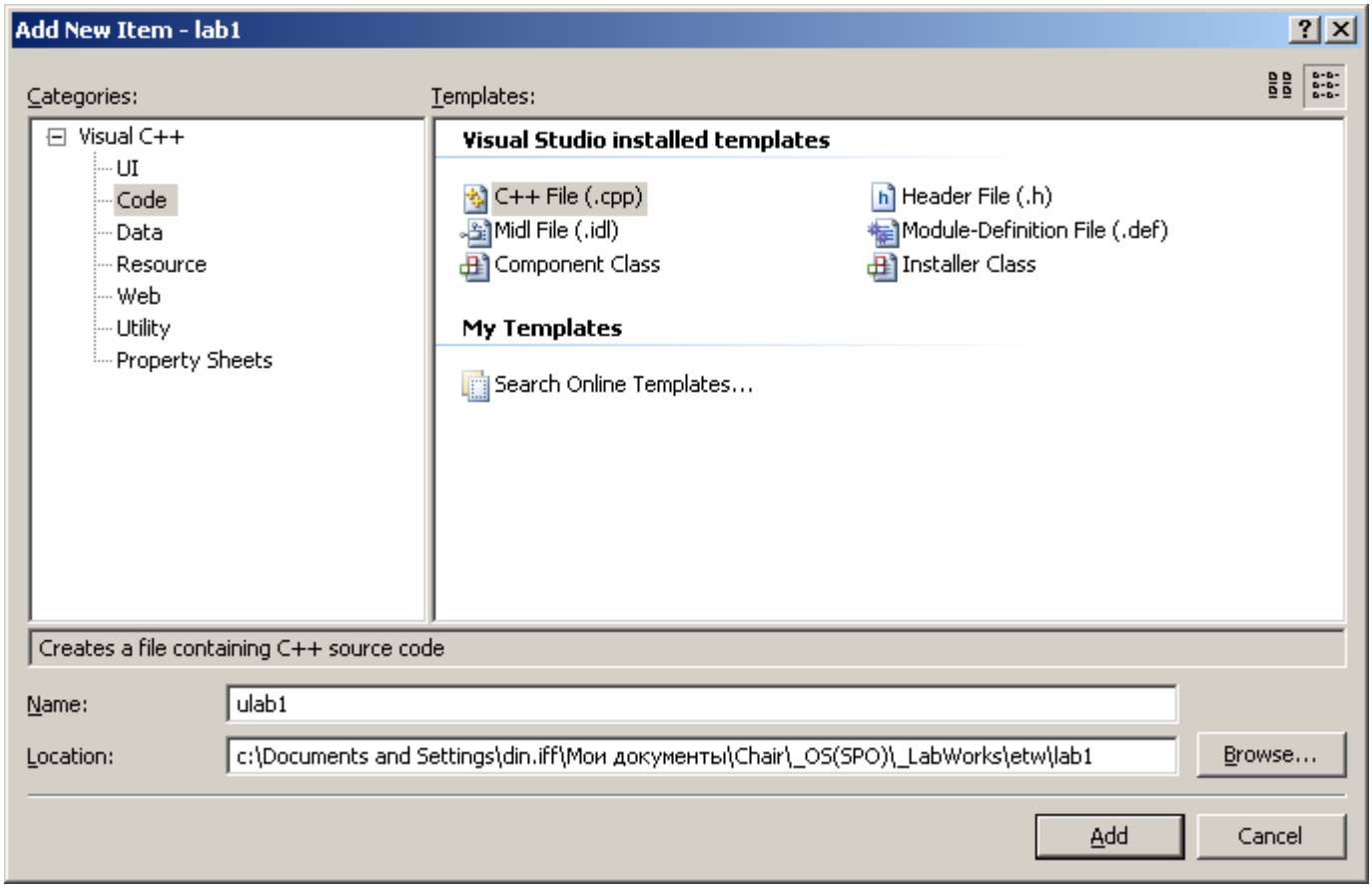


Рис. Б. 7. Диалоговое окно настроек создаваемого объекта

В результате будет создан проект консольного приложения, включающий один модуль исходного кода (пустой) – рис. Б. 8.

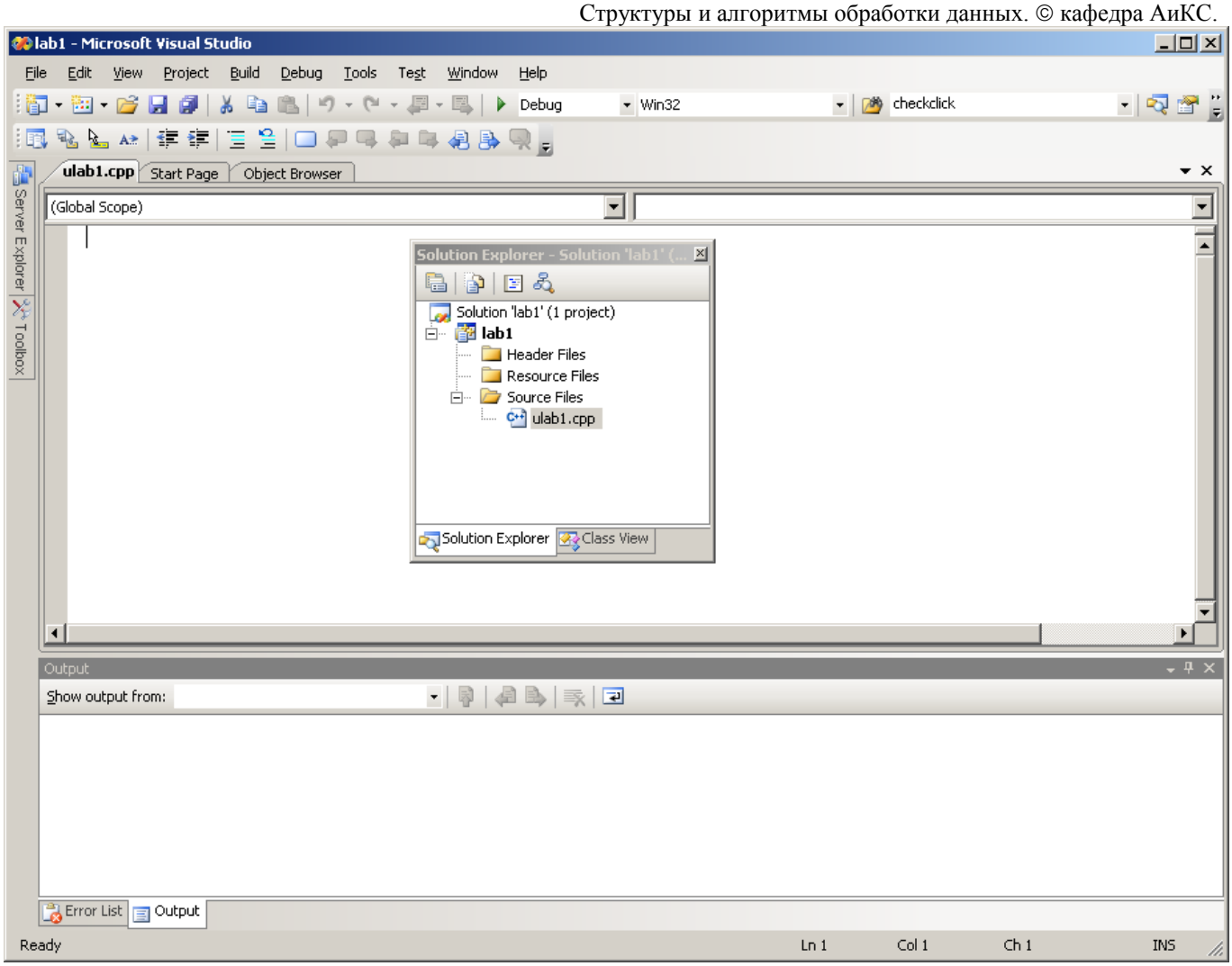


Рис. Б. 8. «Заготовка» проекта консольного приложения