Содержание:

ВВЕДЕНИЕ

С развитием самой первой компьютерной вычислительной техники начал применяться машинный язык для программирования, с помощью которого разработчики того времени могли задавать команды, оперируя при памятью, а также полностью использовать все возможности электронной машины для вычислений разной сложности.

Однако использование многих персональных компьютеров (ПК) на уровне созданного самыми первыми программистами машинного языка очень затруднительно.

Поэтому от его непосредственного применения пришлось отказаться.

На современном этапе также создалось множество языков, что могут могли самые различные функции.

Потом при развитии языков программирования (ЯП), появилась новая возможность для выполнения адаптации обработки информации с использованием созданных программ, что также являлось очень удобным, так как программисты могут создавать программы для определенной предметной области.

Вместе с этим создавались программы с графическим пользовательским интерфейсом, которые имели пользовательское главное или контекстное меню.

Эффективность и необходимость реализации разработки таких программ являются очень актуальными в наше время, поскольку основой для удобного пользования считается дружественный интерфейс для конечного пользователя программой.

Целью исследования является обзор процедур и функций для языков высокого уровня, методики создания пользовательского меню.

Задачи курсовой работы:

– охарактеризовать основные определения теории языков программирования;

- привести основные определения, понятия о применении модульного стиля программирования, как базы для создания пользовательского интерфейса;
- рассмотреть главные принципы создания меню;
- выполнить проектирование функций в C++ Builder для реализации пользовательского интерфейса;
- описать технологии создания оконных программ на практике при использовании пользовательского интерфейса.

Объект работы – ЯП высокого уровня.

Предметом работы являются методы применения процедур, функций для создания пользовательского интерфейса.

По мере возникновения компьютерной вычислительной техники были созданы самые разные методики для написания программ. При этом создавался подход, который помогал многим программистам с растущим усовершенствованием программ.

1. ПОНЯТИЯ О ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

1.1. Понятия и классификация языков программирования

Физические принципы функционирования самых разных электронных устройств типа ЭВМ построены по такому принципу, что компьютер может выполнять разнообразные поставленные команды, состоящие с последовательности единиц и нулей.

На стадии развития вычислительной техники программистам было крайне необходимо составлять свои программные продукты на языках, которые понятны компьютеру, в специальных машинных кодах.

Каждая такая команда могла состоять с кодов для операций, адресов памяти и операндов, которые выражались в виде самых разнообразных сочетаний элементов машинного кода.

Это приводило к острой необходимости поиска средства, которое даст возможности более просто налаживать корректное общение человека и ПК.

Такое средство было в ближайшем будущем найдено: различные ЯП символического типа.

Язык программирования – это искусственный язык, который используется в командном управлении компьютером.

Языки программирования были созданы для того, чтоб обычным пользователям было намного проще создавать программы для ПК, но, при этом, они далее должны были транслироваться (интерпретатором или же транслятором) в некоторый машинный код, что может также быть реализован только лишь компьютером.

Все современные языки программирования можно подразделить на языки высокого и низкого уровня программирования.[4]

Языком низкого уровня часто называют такой язык программирования, предназначенный для определенного типа ПК и может отражать внутренний код.

Также все языки низкого уровня называются машинно-ориентированными языками, ведь их сложно конвертировать на компьютерах с не одинаковыми центральными процессорами, а также довольно сложно изучать, так как для этого крайне необходимо хорошо знать все известные принципы его внутренней работы.[9]

Языком высокого уровня называют такой ЯП, который предназначен для удовлетворения требований программистов, используя словосочетания, что приближены к человеческому языку.

Он никак не может зависит от разных внутренних кодов компьютера и типов процессора. Языки высокого уровня часто применяют для решения проблематики узкого сектора и часто поэтому их называют проблемно-ориентированными.[14]

Отметим, что каждая из команд такого ЯП эквивалентны нескольким командам машинных кодов, поэтому все высокоуровневые программы, намного компактнее, нежели такие же программные продукты, написанные в машинных кодах.[19]

Процесс работы ПК заключен в выполнении последовательности разных инструкций, набора определённых команд, выполняющихся в определённом порядке.

Все машинные виды этих команд, что состоят из последовательности бинарного кода, указывает, какое именно из действий должно выполняться процессором.

Чтоб задать ПК перечень действий, что предназначены для выполнения, нужно задать последовательность с бинарного кода соответствующих программных команд. [3]

Программы, написанные на машинных кодах, могут иметь в своем составе много тысяч команд, а их непосредственное написание – занятие очень сложное.

Намного проще разработать программу с помощью языка, более близкому к естественному языку человека, а саму работу по трансляции программы в машинный код поручить ПК.

Именно так были созданы ЯП, которые предназначены для создания современных программных приложений. [7]

Язык программирования считается формальной знаковой системой, что предназначена для написания кода программ.

Языком программирования может быть определен набор лексических, семантических, а также синтаксических правил, что определяют внешний вид программных средств и действия для реализации их исполнителем (или компьютером).

После создания первых вычислительных программируемых машин разные разработчики придумали более 2100 языков программирования разного уровня, при чем с каждым годом их количество увеличивается. Некоторыми ЯП могут пользоваться только небольшое число программистов, а другие становятся популярными спустя 2-3 года.

Профессиональные программисты могут эффективно применять в своей работе также не менее десятка языков.

Но нет такого программиста, кто бы не мечтал спроектировать свой язык для реализации программного кода: самый быстрый, простой, надежный, удобный. Поэтому за всю долгую историю компьютерной техники люди придумали много ЯП самых разных направлений. [11]

В нынешнее время поставлена перед программистами задача по проектированию системы хранения, обработки некоторых данных, что ещё 20 лет назад были

невозможными.

Так, появляются сверх новые технологии и персональные вычислительные устройства, требующие современных методик к программированию, а развитие Интернета предоставляет новые способы по созданию сетевых технологий обмена информацией.[15]

Всё это может служить некоторой почвой для проектирования новейших ЯП, что могут отвечать реализации всех современных задача использовать новые принципы в программировании, что могут решить проблемы для современных программистов.

Языки очень разные, при этом каждый имеет своё назначение, конкретное направление, уникальную среду по разработке, свои индивидуальные принципы использования и синтаксис. [5]

Сравнительный анализ разных ЯП по возможностям, способам реализации, сложности в освоении является сложнейшей задачей.

Оценивать практическое удобство семантических конструкций можно только на примерах и указывать задачу для всех ЯП, для которой пользоваться лучше определенными из них. [8]

1.2. Язык C++, среда разработки C++ Builder

C++ – ЯП для общего назначения, наиболее часто применим для системного программирования.

Также, ЯП C++ успешно применяется для создания приложений, которые выходят далеко за рамки традиционного системного подхода программирования.

Реализации ЯП C++ есть практически на всех машинах, от самых слабых по мощности – до больших мейнфреймов. [11]

Бьерн Страуструп является проектировщиком языка C++ и главным разработчиком его транслятора.

Также он считается сотрудником компании AT&T. Он получил должность магистра с вычислительной техники в университете Аарус, а докторское звание добыл в университете Кембриджа.

Программист специализируется также и в секторе разработки операционных систем (ОС), различных распределенных систем, а также моделирования и программирования.[17]

Изначально C++ был спроектирован для того, чтобы команде программистов Страуструпа не нужно было создавать программы с помощью ассемблера.

Главным его предназначением было выполнение более приятным процесс программирования и упрощение методологии программирования для отдельно взятых программистов.

Долгое время определенной последовательности разработки для языка C++ не было. Реализация и документирование таких методов шли параллельно.

Такой язык продолжает свое направленное развитие, чтоб преодолеть все возникшие проблемы пользователей.

В 1984 г. стало очевидно, что вся работа по стандартизации ЯП C++ неизбежна и нужно незамедлительно приступить к ней.

Фирма AT&T внесла самый главный вклад в данную работу. Больше ста представителей изучали и комментировали все аспекты ЯП, что стали современной версией по руководству и материалами для описания стандарта C++. [3]

При разработке C++ самыми важнейшими критериями были простота и удобство программирования. При возникновении вопроса, что именно упростить: руководство либо другую документацию по ЯП или же транслятор, то выбирали постоянно первое. Также огромное значение имела совместимость с языком C.[9]

Ключевое понятие ЯП C++ – это класс – определяемый пользователем формат данных.

Он обеспечивает преобразование, инициализацию, динамическое определение разных типов, которое часто может контролироваться пользователем, а также управление памятью, средствами для перегрузки функций.

В C++ реализованы полнее концепции модульного построения программ и контроля типов данных.

C++ содержит также некоторые усовершенствования, он может определять разные стандартные значения параметров, функции-подстановки, операции

управления памятью, реализацию перегрузки имен функций, а также применение ссылочного типа.

Среда C++ Builder является выпущенным средством для разработки программных приложений, что позволяет также создавать приложения с графическим интерфейсом при помощи ЯП C++, используя библиотеку визуальных компонентов Delphi. [7]

C++ Builder – это SDI-приложение, главное окно имеет настраиваемую панель и палитру (рисунок 1).

При запуске среды отображаются окно инспектора объектов, формы новых приложений. Ниже, под окном для пользовательской формы находится область написания кода.

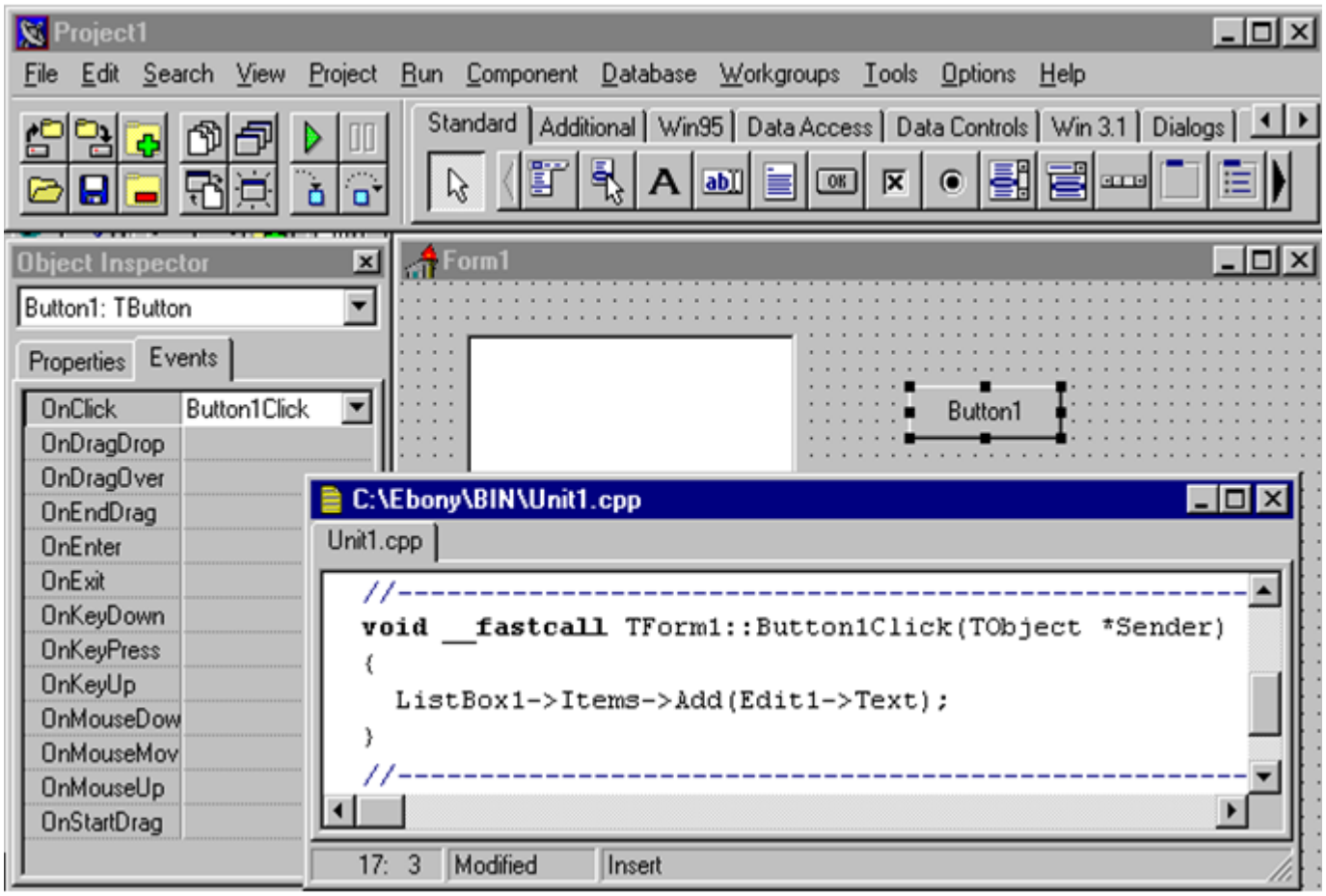

Рис.1. Программный интерфейс

Формы – это база всех приложений в рассматриваемой среде. Для создания пользовательского интерфейса надо добавить в форму компоненты объектов.

Компоненты разделяются на такие категории:

– визуальные;

– невизуальные.

Визуальные составляющие появляются во время запуска программы, а невизуальные будут отображаться лишь во время разработки, и их не видно уже при выполнении программы (рисунок 2).

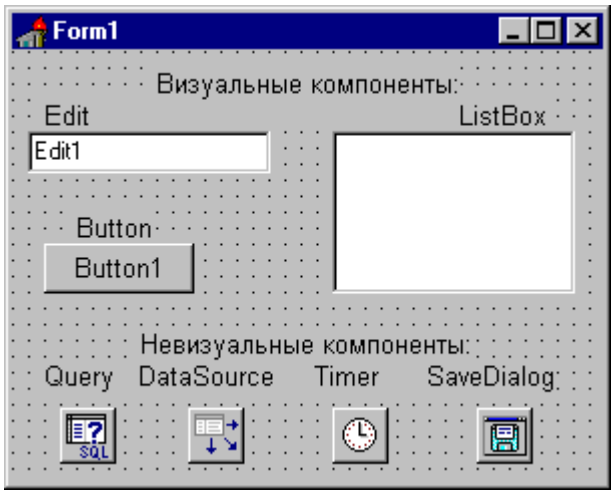

Рис.2. Образцы компонентов

Каждый компонент имеет основные 3 характеристики: событие, свойство, метод.[15]

Если выбрать компонент на палитре, добавить его на форму, то инспектор отобразит автоматически события, а также свойства, что могут использоваться (рисунок 3).

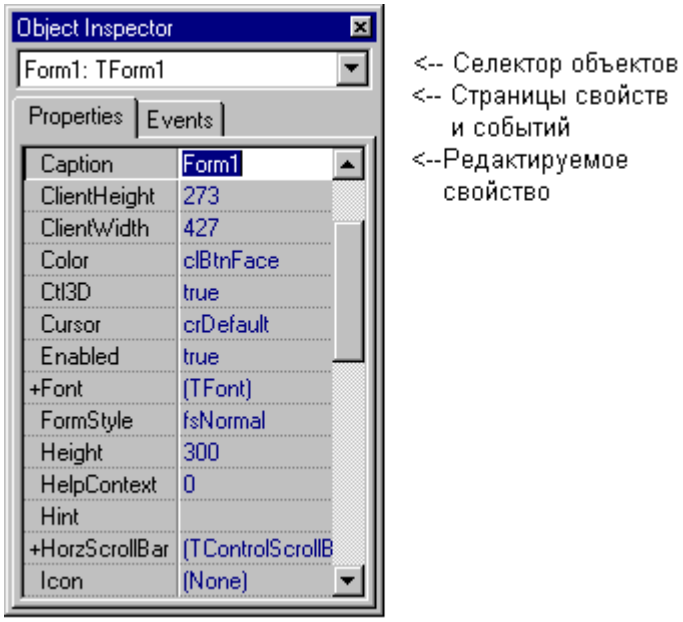

Рис.3. Инспектор объектов

Свойства компонентов – это их атрибуты, определяющие его поведение, а также внешний вид.

В первом разделе работы рассмотрены основные понятия по теории языков программирования, а также приведены определения языка программирования, классификация (языки низкого, высокого уровней).

Также рассмотрены все основные понятия популярного языка программирования C++, среды интегрированный разработки программных продуктов C++ Builder.

2. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ. ПРИМЕНЕНИЕ ПОЛЬЗОВАТЕЛЬСКИХ МЕНЮ

2.1. Основы модульного программирования в C++

Когда программа использует большой размер и становится сложной для восприятия, есть резон разделять ее на несколько небольших логических по содержанию функций, из которых каждая выполнять будет свою определенную задачу.[20]

В результате программы будут более структурированными и легкими для понимания.

Кроме того, при разработке функций не требуется создания одинаковых фрагментов кода.

Непосредственное разделение программы на несколько функций – это базовый принцип для структурного программирования.

Функция – это специальная именуемая независимая часть программного продукта, что может многократно вызываться из другой части программного продукта, манипулировать информацией и выполнять возвращение полученных результатов.

Все функции имеют также собственное имя, по которому осуществляется ее вызов.

В программировании различают 2 виды функций:

– встроенные стандартные функции (sin(), sqrt (), …);

– функции, что создаются пользователями для собственных потребностей. [18]

Создаваемая функция в программе должна быть объявленной и определенной.

Непосредственное объявление функции должно быть описано в программе раньше за ее непосредственное использование.

Определение может находиться в самых разных местах программы, кроме тела других функций. [20]

Объявление функции содержит также прототип и имеет такую форму:

<тип_результата> <имя > (<список аргументов и типов данных>)

Результат возвращается при помощи оператора return. Прототип для функции также может находиться в программе и непосредственно в заголовочных файлах.

Также, кроме объявления, все спроектированные функции должны иметь свое определение.

Определение функции содержит в себе тело функции – команды, что выполняет непосредственно функция, и команды для возврата результата (в С++ команда return) [1]:

<тип_функции> <имя> (<список_параметров>)

{

<тело >

return <результат>

}

Для определения, к примеру, среднего значения 2-х чисел, определение подпрограммы sr будет таким: [6]

float sr(float a1, float b1)

{

float s;

s = (a1 + b1)/2;

return s;

}

Приведенная функция имеет возможность вычислить среднеарифметическое для двух действительных чисел a1, b1 и в точку вызова передать его значение.

При этом, в случае, в котором определение функции размещается в программе ранее точки для вызова, писать отдельно объявления и саму реализацию функции необходимости нет, объявление функции можно также избегать. [6]

Если функции выполняют при этом вычисления и действия, не требующие возвращение результатов, то их тип указывают как void (пустой тип).

Для таких функций может отсутствовать оператор return или не иметь значения, которое возвращается.

Самым основным примером таких функций считаются отзывы событий в C++ Builder:

void __fastcall TForm1 :: Buton1Click (TObject * Sender)

{

```
}
```

В ниже представлено на примере создание собственной функции для вычисления и поиска наибольшего с 3 действительных чисел, вывод результатов вычислений с помощью оператора return:

```
void m3(float x_1, float y_1, float z_1)

{

if (x_1> y_1 && x_1> z_1)

cout << x_1;

else

if (y_1> x_1 && y_1> z_1)

cout << y_1;

else

cout << z_1;

}
```

Выполнение функций реализуется в случае, когда в коде программы может быть встречен оператор вызова, то есть имя функции с четко определенными параметрами.

При этом формальные параметры последовательно принимают значения фактических.

Важными правилами являются количество, порядок записи, соответствие разных типов аргументов, иначе это все может приводить непосредственно к серии ошибок.

Если же функции объявлены с самым разным типом данных, кроме типа void, то при вызове ее нужно присваивать значение переменной аналогичного типа, что и функция. [16]

Например, функцию sr() можно вызвать:

int a_1 = 5, x_1 = 2, y_1 = 3, z_1, s_1, s_2;

1) z_1 = sr (5, 4); // Z_1 = 4.5

2) s_1 = sr (a1, 11); // S_1 = 8

3) s_2 = sr (x1, y1) // S_2 = -0.5

Рекурсивной называется функция, выполняющая вызов самой себя. Указанная рекурсия считается прямой.

Также существует и называемая косвенная рекурсия, для которой две или же больше функций вызывают несколько раз саму себя.

Понятие рекурсии иллюстрируется матрешками. Это также значит, что понятие рекурсии может свести сложную задачу к реализации ее уменьшенных копий. [18]

Последовательно уменьшая размерность указанных задач, а рекурсия в конце концов приводит к известному решению, также применяя который можно получать легко решение начальных задач.

Более классические примеры реализации рекурсии находятся в математике, поскольку рекуррентные соотношения помогают определять элемент последовательности. К примеру, рекурсивное соотношение в значениях Фибоначчи:

F (m) = F (m-1) + F (m-2),

тут F (0) = F (1) = 1.

Если же провести рассмотрение указанной последовательности, начиная с младших к старшим компонентам, способ ее построения задасться циклическим алгоритмом, но в случае наоборот – с заданного n, а методика определения через предыдущие будет рекурсивной.[10]

Очевидно, что рекурсия не может являться безусловной, ведь в указанном случае она была бы бесконечной. Рекурсия внутри имеет условие завершения своего алгоритма, по которому каждый следующий шаг уже не будет выполняться.

Рекурсивные функции на первый взгляд лишь похожи на классические куски программ.

Внешне создается впечатление, что текст функции воспроизводится каждый раз.

На самом деле эффект воспроизводится в ПК. Но копируется вовсе не весь при этом текст из функции, а некоторые ее части, связанные с локальными данными (фактические и формальные параметры, локальные значения, точка поворота). [19]

Алгоритмическая часть программы (разные операторы, выражения) для рекурсивных функций, глобальные переменные не изменяются вовсе, они присутствуют в единственных экземплярах.

Все вызовы рекурсии порождают новый "экземпляр" специальных параметров, локальных переменных, а при этом старые "экземпляры" далеко не уничтожаются, а просто сохраняются в стеке.

Здесь имеет место случай, когда одному и тому же имени переменной в работе программы соответствуют определенное количество ее экземпляров. [18]

Указанные переменные имеют возможность создать группу (фреймы стека). Сам стек "запоминает историю" всех вызовов в качестве последовательности фреймов.

Программа для конкретного момента работает непосредственно с последним вызовом, фреймом.

Для завершения процесса рекурсии программа может возвратить к предыдущей версии рекурсивно выполненной функции, а также к предыдущему фрейму в программном стеке.

2.2. Принципы создания пользовательского интерфейса

Создание качественного интерфейса требует большего, чем просто соблюдение простых инструкций. Оно предполагает при реализации принцип, который определяется так: «интересы пользователя являются превыше всего», соответствующую методологию разработки ПО.

Главное достоинство интерфейса программы заключается в том, что все пользователи должны чувствовать, что они управляют ПО, а не ПО управляет ими.

Для вызова у пользователя этого ощущения так называемой «внутренней свободы» каждый программный интерфейс должен обладать свойствами.

Естественный интерфейс – интерфейс, что не вынуждает пользователей изменять существенно привычные способы решения задач.

В частности, сообщения, а также полученные результаты, выдаваемые приложением, не должны вовсе требовать каких-то дополнительных разъяснений.

Согласованность программного интерфейса имеет возможность для пользователей перенести имеющиеся знания на новые задания, осваивать разные аспекты быстрее, благодаря этому фокусировать внимание на решаемых проблемах, а не тратить свое время для уяснения применения компонентов управления.

Дружественность интерфейса. Пользователи обычно изучают практически все возможные особенности обработки данных с новыми программными продуктами. [2]

Эффективный интерфейс должен принимать постоянно во внимание именно указанный подход. На таком каждом шаге работы должен он разрешить только определенный набор действий и предупреждать пользователей о внештатных ситуациях, где все они могут вредить программе или же данным.[4]

Эстетическая привлекательность интерфейса. Реализация визуальных компонентов является важнейшей частью для разработки современного программного интерфейса.

Корректное представление всех используемых визуальных объектов обеспечивает очень хорошую передачу информации о поведении, взаимодействии компонентов.

В это же время надо помнить, что все использованные визуальные элементы, появляющиеся на экране, потенциально будут требовать внимания пользователя.

Меню является популярным вариантом для организации разного рода запросов по вводу информации в программу.

Существует несколько форматов представления пользовательского меню:

- ○ список объектов;
  ○ меню в виде блоков;
  ○ меню в виде строк;
  ○ меню в виде пиктограмм.

Меню в виде строк данных может появляться вверху экрана и часто остается на этой позиции на протяжении всего пользовательского диалога. Таким образом, помощью меню удобно отображать разные варианты данных для ввода.

Меню можно с равным успехом применять для ввода как управляют их сообщений, так и данных. Приемлемая структура меню зависит от его размера и организации, от способа выбора пунктов меню и реальной потребности пользователя в поддержке со стороны меню.

Во втором разделе курсовой работы рассмотрены основные понятия и принципы модульного программирования, описаны понятия функций, виды функций в C++, рассмотрен принцип реализации рекурсивных функций.

Также рассмотрены все основные принципы для создания пользовательских меню.

3. ПРАКТИЧЕСКОЕ ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ И СОЗДАНИЕ ИНТЕРФЕЙСА В C++ BUILDER

3.1. Проектирование функций в C++ Builder

Рассмотрим использование пользовательских функций по примеру разработки программного продукта по демонстрации касательной в движении по графику функций.

Рассмотрим интерфейс программы:

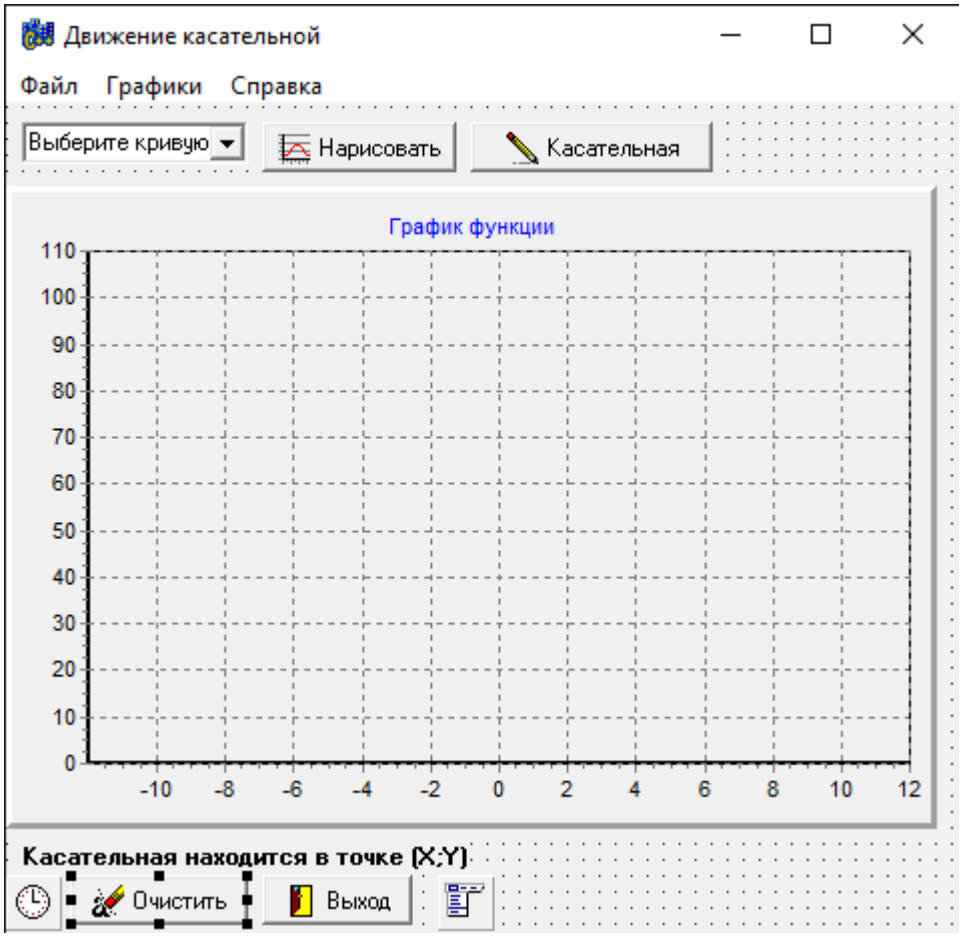

Рис.4. Пользовательский

На форме помещены компоненты:

– Кнопка «Нарисовать»;

– Кнопка «Касательная»;

– Кнопка «Очистить»;

– Кнопка «Выход»;

– выпадающий список с функциями;

– компонент для создания графика;

– надпись о точке соприкосновения непосредственно с касательной;

– таймер;

– главное меню.

Для работоспособности создаваемой программы, кроме событий, созданы такие подпрограммы:

– функция очистки данных с полей:

```
void clear()

{

//установка таймера

Form1->Timer1->Enabld=false;

//установка флага

o_f=false;

//очистка серии 3

Form1->Series3->Clear();

//очистка серии 2

Form1->Series2->Clear();

//очистка серии 1

Form1->Series1->Clear();

//удаление значений выпадающего списка ComboBox1

Form1->ComboBox1->Clear();

//ввод данных выпадающего списка по умолчанию

Form1->ComboBox->Text="Укажите кривую";

//начальные настройки диаграммы

Form1->Chart->Title-> Clear();

//добавление слова «График»

Form1->Chart->Title-> Add("График ");
```

```
//индикаторы для положения касательной

Form1->Label->Caption="Касательная";

}
```

Также рассмотрим пользовательскую функцию, при помощи которой создается график.

```
//название функции

void __fastcall TForm1::BitBtn1Click(TObject *Sender)

{

// очистка поля

Chart1->Title-> Clear();

//очистка серии 3

Series3->Clear();

//очистка серии 2

Series2->Clear();

//очистка серии 1

Series1->Clear();

//остановка таймера

Timer1->Enabld=false;

//индикатор функции в списке

if(ComboBox->ItemIndex==0)

o_f=false;

else

o_f=true;
```

```
// график функции синус
if(ComboBox->ItemIndex==0)
{
Chart1-> Add("График синуса");
for(x=-7;x<=7;x+=0,1)
{
y=sin(x);
Serie1->Add (x, "",clBlue);}
Size(1.3,-1.4,7,-6,9);
x1=-6;
}
else
//создание графика косинуса
if(ComboBox->ItemIndex==1)
{
Chart1-> Add("Косинус");
for(x=-7;x<=8;x+=0.1)
{
y1=cos(x);
Series->AddXY(x,clGreen);}
Size(1.2,-1.4,7,-6,9);
x1=-4,9;
}
```

```
else

// функция exp

if(ComboBox->ItemIndex==2)

{

for(x=-6;x<=3;x+=0.1)

{

y=exp(x1);

Series1->AddXY(y,clRed);}

Chart1->Title->Add("Экспонента");

Size(7,-1,3,-7);

x1=-6;

}

else

// если не выбрана функции

if(ComboBox->ItemIndex==-1)

ShowMesage("Выберите функцию");
```

3.2. Создание меню в C++ Builder

Рассмотрим процесс проектирования меню пользователя для создаваемой программы.

Для создания меню пользователя нужно добавить специальный невизуальный компонент под названием MainMenu1 на проектируемую форму.

Главное свойство компонента рассматриваемого компонента Items. Заполнение производится при использовании Конструктора Меню, что также вызывается двойным нажатием по элементе MainMenu или же двойным нажатием кнопки с

троеточием около свойства под названием Items в инспекторе.

В этом окне также присутствует возможность для проектирования всего пользовательского меню.

Для работы непосредственно в конструкторе меню се новые разделы можно вводить с помощью помещения курсора в рамке с точек.

Свойство под названием Caption дает возможность для создания надписи раздела.

Заполнение свойства является аналогичным как и для процесса заполнения такого же свойства на кнопках, при этом включая применение символа амперсанда для реализации комбинации клавиш по быстрому доступу.

Если же значение свойства Caption очередного раздела будет равно символу минус "–", то вместо такого раздела в меню пользователя появится разделитель.

Свойство под названием Name указывает на имя объекта, который соответствует определенному разделу меню.

Свойство Shortcut имеет возможность для определения клавиш быстрого доступа к каждому из разделов меню, а именно разные «горячие» клавиши, при использовании которых пользователь, даже не заходя даже в меню, может вызывать в любой момент процедуры, что связанные по данному разделу программы.

Чтобы определить такие клавиши для быстрого доступа к меню, нужно открыть имеющийся выпадающий список для свойства Shortcut непосредственно в окне инспектора объектов и выбрать необходимую комбинацию клавиш с него. Эта комбинация будет сразу показана на строке меню.

Свойство под названием Default определяет, будет данный раздел разделом по умолчанию для указанного подменю (разделом, куда вход выполняется двойным нажатием на родительском разделе).

Подменю может содержать один раздел по умолчанию, что выделяется полужирным шрифтом.

В среде C ++ Builder 6 введено новое свойство для работы с разделами меню – AutoCheck.

Если же его установить по значению true, пользователем при каждом выборе указанного раздела маркер будет переключаться автоматически, указывая то на выбранное пользователем состояние, то на отсутствие его.

Нажав дважды на компонент откроется окно для редактирования меню, куда и нужно ввести информацию (рисунок 5):

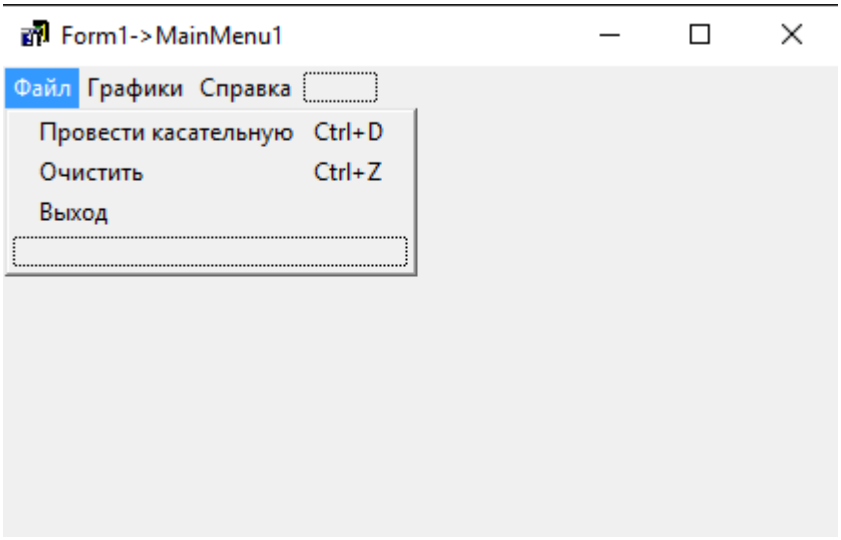

Рис.5. Проектирование меню

Получим в результате пункты (рис.6):

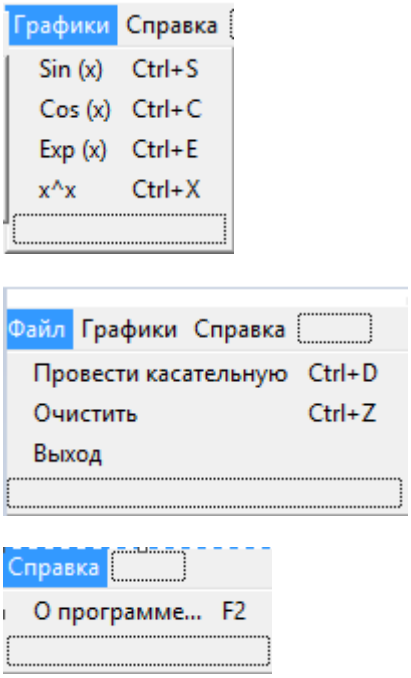

Рис.6. Пункты главного меню

Продемонстрируем функционирование созданной программы (рисунок 7):

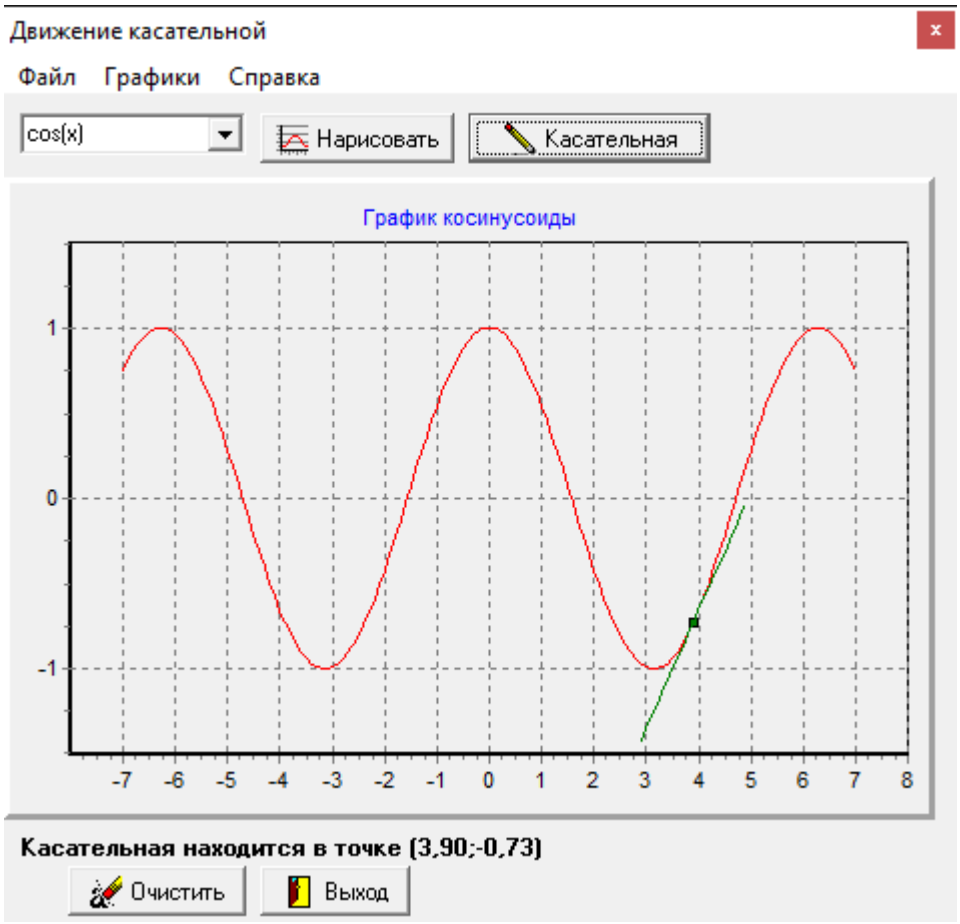

Рис.7. Движение касательной

3.3. Разработка пользовательского интерфейса в консольных приложениях

Рассмотрим применение пользовательского меню для реализации модульной программы следующего содержания:

В среде программирования C++ разработать программу из меню пользователя, которая над введенным массивом выполняет следующие операции: [9]

1. Выводит массив на экран;

2. Транспонирует массив;

3. Находит попарно сумму элементов массива (первый элемент + последний, второй + предпоследний и т.д.);

4. Удаляет элемент, который находится на позиции с номером X;

5. Сортирует каждые Y элементов на росте.

6. Записывает начальный массив в текстовый файл;

7. Формирует с одномерного массива квадратную матрицу минимального размера (в случае отсутствия необходимого количества элементов добавить значение 0).

Программа будет написана в среде разработки Dev-C++ на соответственном ЯП высокого уровня.

Рассмотрим функции, которые будут применяться для организации программы: [3]

– dim() – функция предназначена для ввода размерности используемого массива;

– input() – функция предназначена для ввода элементов массива при использовании генератора псевдослучайных чисел;

– output() – функция для организации вывода элементов массива;

– sum() – функция для нахождения суммы элементов массива;

– sort() – функция для сортировки элементов массива методом пузырька;

– sh() – вывод искомого элемента;

– vs() – функция, реализующая вставку элемента;

– rez() – функция для удаления элемента массива;

– menu() – функция для реализации меню программы;

– main() – главный модуль программы.

Основная функция, которая применяется для вывода меню выводит перечень пунктов меню в текстовом режиме (рисунок 8):

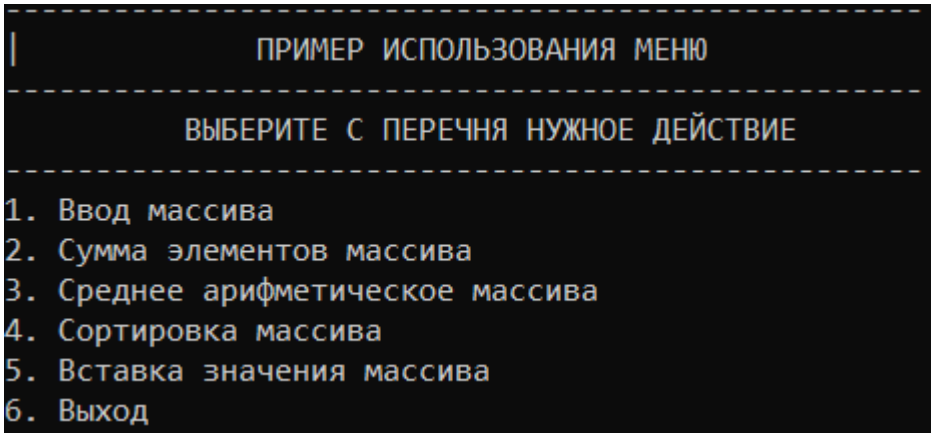

Рис.8. Пример меню

В зависимости от выбора значения (1, … , 6) будет реализован соответствующий пункт меню. [16]

Для создания такого меню нужно ввести программный код, показанный в приложении 1.

Стоит отметить, что количество пунктов меню легко увеличивается в результате добавления новой строки с оператором вывода cout. [12]

Логика программы построена таким образом, что при выборе соответствующего пункта будет запущена одна с функций.

Такая возможность реализована с помощью оператора множественного выбора switch.

Непосредственно в операторе будет анализироваться переменная q, которая хранит введенное при выборе пункта меню значение.

Оператор case проверяет, какое именно значение введено, и в зависимости от него запускается определенная подпрограмма: [8]

switch(q)

{

//выбор значения 1

case 1:

//запуск функции для вывода массива

```
output(a,n);
//задержка на экране
system("pause");
//очистка экрана
system("cls");
//выход с оператора выбора
break;
//аналогично выполняются остальные составные части оператора выбора[4]
case 2:
//вывод суммы элементов
cout<<"Сумма: "<<sum(a,n)<<endl;
system("pause");
system("cls");
break;
//вывод среднего значения
case 3:
cout<<"Среднее: "<<sum(a,n)/n<<endl;
system("pause");
system("cls");
break;
//сортировка массива
case 4:
sort(a,n);
```

```
system("pause");
system("cls");
break;
//вставка элемента массива
case 5:
x=sh();
y=vs();
rez(a,n,x,y);
system("pause");
system("cls");
break;
//выполнение закрытия программы
case 6:
cout<<"Программа окончена"<<endl;
system("pause");
break;
//break;
//случай некорректного ввода данных
default:
cout<<"Введите корректное значение";
getch();
system("cls");
}
```

Рассмотрим подробнее некоторые функции, которые применены в рассматриваемой программе.

Функция для ввода размерности одномерного массива:

```
//определение функции

int dim()

{

//объявление переменной функции

int n;

//вывод сообщения

cout<<"Размерность: ";

//ввод размерности

cin>>n;

//возврат значения из функции

return n;

}
```

Функция для нахождения суммы элементов массива[20]

```
//название прототипа функции

int sum(int a[], int n)

{

//инициализация переменной

int s=0;

//нахождение суммы массива

for(i=0;i<n;i++)
```

```
s+=a[i];

//возврат значения

return s;

}
```

Другие функции созданы по аналогии.

Рассмотрим пример использования данной программы.

После запуска нужно выполнить ввод размерности. Автоматически будет сгенерирован массив и будет показано меню пользователя (рисунок 9):

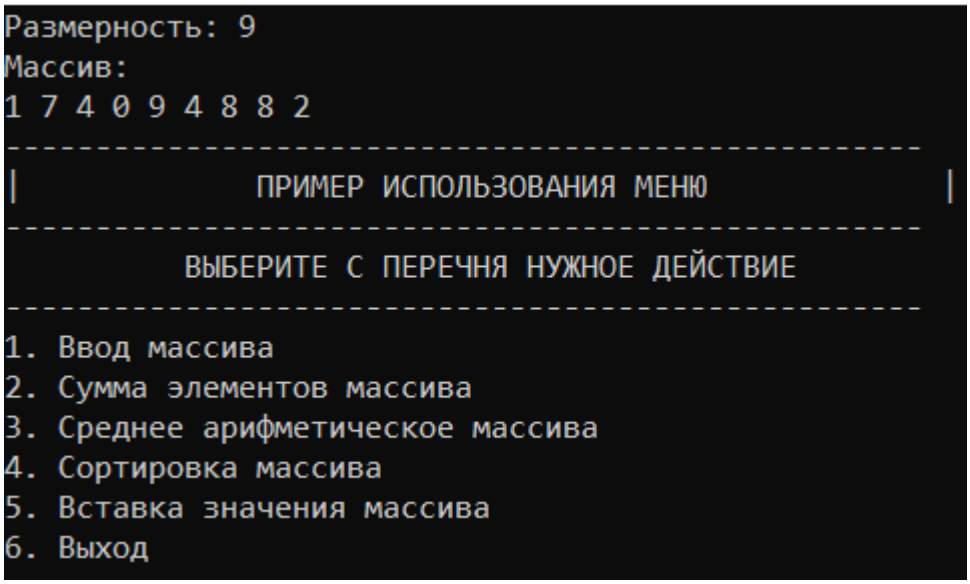

Рис.9. Ввод размерности

Для, к примеру, вывода сортировки массива, нужно выбрать пункт 4. В результате получим (рисунок 10):[15]

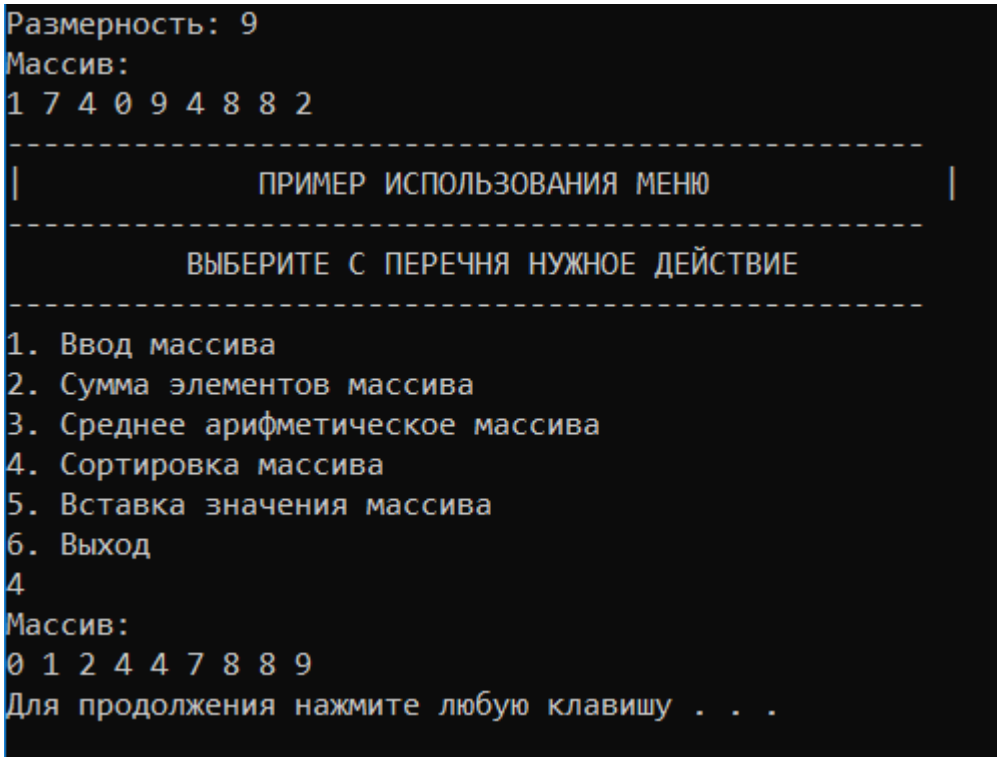

Рис.10. Результат сортировки

После нажатия любой клавиши опять будет показано меню с выводом его пунктов.

В последнем разделе курсовой на практике описаны все методы построения меню, применение функций пользователя при проектировании программных продуктов.

ЗАКЛЮЧЕНИЕ

В настоящее время, в котором значительно усиливается связь между секторами коммерции и разработки ПО, а также в процессах, когда корпорации тратят много усилий на планирование своего бизнеса, ощущается острая надобность в соответствии имеющихся бизнес-процессов и их реализаций.

Но, к сожалению, огромное число языков не имеют прямого пути для использования их в описании бизнес-процессов.

Например, на сегодняшний день разработчики выполняют комментарии своих программных продуктов для объяснения того, какие объекты реализуют определенный абстрактный бизнес-компонент. Язык C++ позволяет также использовать расширяемые, типизированные данные, которые часто бывают прикреплены к объекту.

Непосредственно архитектурой проекта могут определяться локальные атрибуты, что также будут связанны с многими элементами ЯП – классами, интерфейсами.

В процессе выполнения курсовой работы были получены, закреплены практические навыки по реализации разработки программ при использовании модульного программирования, а также в написании программы в среде C++ Builder.

В работе были реализованы следующие задачи:

– охарактеризовать основные определения теории языков программирования;

- привести основные определения, понятия о применении модульного стиля программирования, как базы для создания пользовательского интерфейса;
- рассмотреть главные принципы создания меню;
- выполнить проектирование функций в C++ Builder для реализации пользовательского интерфейса;
- описать технологии создания оконных программ на практике при использовании пользовательского интерфейса.

В процессе анализа имеющейся информации выявлены такие достоинства программ с применением пользовательского интерфейса:

- удобность ввода данных;
- структурированное отображение информации;
- создание дружественного интерфейса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аксёнова Е.А. Алгоритмы и структуры данных на C++: Учебное пособие. - Петрозаводск: Изд-во ПетрГУ, 2016.- 81 с.
2. Беляева, И. В. Основы программирования на языке Turbo Pascal: учебное пособие / И. В. Беляева. - Ульяновск: УлГТУ, 2015. - 266 с.
3. Бобровский С. Самоучитель програмирования на языке C++ в среде Borland C++ Builder М.: ИНФРА-М, 2013.–251 с.
4. Бруно Бабэ. Просто и ясно о Borland C++: Пер. с англ. - Москва: БИНОМ, 2014. – 400с.

5. Заварыкин В.М.. Основы информатики и вычислительной техники. — М.: Просвещение, М.: - 2014. 180 с.
6. Задачи по программированию / С. А. Абрамов. — М.: Наука, 2014. – 344 с.
7. Задачник-практикум: В 2 т. / Под ред. И. Г. Семакина. — М.: Лаборатория Базовых Знаний, 2013. – 182 с.
8. Зубов В. С. Программирование на языке С++. — М.: Информационно-издательский дом «Филинъ», 2013. – 452 с.
9. Зуев Е. А. Практическое программирование на языке С++. — М.: Радио и связь, 2014. – 406 с.
10. Йенсен К. С++ — руководство для пользователей и описание языка. — М.: Мир, 2015. – 402 с.
11. Касаткин В. Н. Информация. Алгоритмы. ЭВМ. — М.: Просвещение, 2015. – 210 с.
12. Керниган Б. Язык программирования Си: Пер. с англ. — М.: Финансы и статистика, 2014. – 322 с.
13. Культин Н.Б. Визуальное программирование.— СПб.: BHV — Санкт-Петербург, 2013. – 266 с.
14. Культин Н.Б. С++ Buider в задачах и примерах – СПб.:БХВ-Петербург, 2015. – 336с.: ил.
15. Липачев Е.К. Технология программирования. Базовые конструкции С/С++ / Е.К. Липачев. - Казань: Казан. ун-т., 2015. - 142 с.
16. Мамонова Т.Е. Информатика. Общая информатика. Основы языка С++: учебное пособие / Т.Е. Мамонова; Томский политехнический университет. - Томск: Изд-во Томского политехнического университета, 2014. - 206 с.
17. Мамонова Т.Е. Информатика. Программирование на С++: учебно-методическое пособие / Т.Е. Мамонова; Томский политехнический университет. - Томск: Изд-во Томского политехнического университета, 2013. - 118 с.
18. Марапулец Ю.В.Программирование на языке высокого уровня: Учебное пособие. - Петропавловск-Камчатский: КамчатГТУ, 2015. - 189 с. ISBN/ISSN:978-5-328-00185-4
19. Объектно-ориентированное программирование на С#: Учебное пособие / А.А. Андрианова. - Казань: Казанский (Приволжский) федеральный университет, 2015. - 134 с.
20. Шамшев, А.Б. Алгоритмическое мышление при решении задач (на примере языка С#): учебное пособие / А.Б. Шамшев, К.В. Святов. - Ульяновск: УлГТУ, 2014. - 113 с.

ПРИЛОЖЕНИЯ

Листинг программного кода для описания меню

```
int menu()

{

int q;

cout<<"-------------------------------------------------"<<endl;

cout<<"| ПРИМЕР ИСПОЛЬЗОВАНИЯ МЕНЮ |"<<endl;

cout<<"-------------------------------------------------"<<endl;

cout<<" ВЫБЕРИТЕ С ПЕРЕЧНЯ НУЖНОЕ ДЕЙСТВИЕ "<<endl;

cout<<"-------------------------------------------------"<<endl;

cout<<"1. Ввод массива"<<endl;

cout<<"2. Сумма элементов массива"<<endl;

cout<<"3. Среднее арифметическое массива"<<endl;

cout<<"4. Сортировка массива"<<endl;

cout<<"5. Вставка значения массива"<<endl;

cout<<"6. Выход"<<endl;

cin>>q;

return q;

}
```