

Большаков С.А. , к.т.н., доц. Каф. ИУ5

**Методические указания к лабораторной работе № 1
по курсу ОСНОВЫ ПРОГРАММИРОВАНИЯ
ГУИМЦ**

**"Среда программирования MS VS. Программные проекты. Линейные программы.
Вычисления"
(4 часа)**

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	2
1. Цель лабораторной работы №1 по дисциплине Основы программирования - ОП.....	4
2. 2. Порядок выполнения лабораторной работы.....	4
3. 3. Основные понятия.....	4
3.1. 3.1 Линейные вычислительные процессы.	4
3.2. 3.2 Пример простейшей программы на языке СИ.	5
3.3. 3.3 Программные проекты в системах программирования на СИ.	5
3.4. 3.4 Создание консольного программного проекта в СП на СИ.	5
3.5. 3.5 Русификация проекта консольного ввода и вывода.	6
3.6. 3.6 Программы и программирование.....	7
3.7. 3.7 Переменные	8
3.8. 3.8 Константы	8
3.9. 3.9 Операторы и составные операторы	9
3.10. 3.10 Первая программа и русификация	10
3.11. 3.11 Подключение функций математической библиотеки.	11
3.12. 3.12 Программа вычисления синуса.	12
3.13. 3.13 Форматированный ввод-вывод.....	12
3.14. 3.14 Вывод данных – функция printf.....	13
3.15. 3.14 Ввод данных – функция scanf.....	14
3.16. 3.16 Отладка программ.	15
3.17. 3.17 Пример программы линейного вычисления с вводом и выводом данных.	15
3.18. 3.18. Болванка для главного модуля с математической библиотекой	17
3.19. 3.19. Болванка для демонстрации ЛР или работы на семинаре	17
3.20. 3.20. Болванка с меню, выходом и возвратом в меню	19
3.21. 3.21. Болванка с меню, выходом и возвратом в меню для ДЗ и выходом	20
4. 4. Контрольные задания ЛР №1.....	23
4.1. 4.1 Изучить теоретическую часть ЛР	23
4.2. 4.2 Изучить создание проектов и создать первый проект (First).....	23
4.3. 4.3 Выполнить русификацию проекта.	23
4.4. 4.4 Подготовить и выполнить первую простейшую программу по шагам.	23
4.5. 4.5 Изучить и выполнить подключение математической библиотеки в проект и вычислить выражение, подключить константу Пи (π) и вычислить синус для выражения.....	23
4.6. 4.6 Изучить и проверить работу функций ввода и вывода.	23
4.7. 4.7 Подготовить и выполнить в режиме отладки линейную программу с вводом, расчетом и выводом.	24
4.8. 4.8 Подготовить и выполнить программу по своему варианту.	24
4.9. 4.9 Подготовить и выполнить программу с дополнительными требованиями	24
4.10. 4.10 Оформить отчет по ЛР по представленному шаблону (в архиве с МУ ДР №1 на сайте) и с указанными рекомендациями (там же).	24
5. 5. Контролируемые требования ЛР №1.	24
6. 6. Варианты заданий для п.п.4.8 Задания ЛР №1.	25
6.1. 5.1 Демонстрация и защита.	26
6.2. 5.2 Продемонстрировать работу пунктов ЛР преподавателю (демонстрация отмечается в журнале ЛР).	26
6.3. 5.3 Подготовить и защитить отчет по ЛР.....	26
6.4. 5.4 Прочитать и подготовить ответы на контрольные вопросы.....	26
6.5. 5.5 Выполнить программу с дополнительными требованиями.....	26
7. 7. Дополнительные требования для студентов СУЦ (д.т.).	26
8. 8. Демонстрация, защита ЛР и отчет по ЛР.....	28

ОП ГУИМЦ 2024 ЛР№1

9. 9. Срок демонстрации и защиты.	29
10. 10. Контрольные вопросы по ЛР.	29
11. 11. Литература.	29
12. 12. Приложение:	30
12.1. Математическая библиотека	30

1. Цель лабораторной работы №1 по дисциплине Основы программирования - ОП

Целью данной ЛР по дисциплине ОП является получение начальных знаний и умений для программирования линейных программ на языке программирования СИ. Студенты осваивают технологию создания простого консольного проекта и отлаживают в среде программирования MS VS 2005/2008/2010 простую программу, содержащую вычисления и операции ввода и вывода данных средствами библиотеки **RTL** (**RunTimeLibrary**). Они используют в системе программирования программный проект, создают и настраивают его для своей задачи. Студенты знакомятся с возможностями системы программирования, возможностями вычисления выражений, вывода результатов работы программы, а также выполняют отладку программы и генерируют исполнимую программу, готовую к выполнению на компьютере. Они учатся оформлять отчет по ЛР и защищать работу.

2. 2. Порядок выполнения лабораторной работы

1. Познакомиться с содержанием методических указаний и основными понятиями ЛР (разделы 3 и 4)
2. Проработать порядок выполнения работы (раздел 5).
3. Создать консольные проекты для проверки примеров и выполнения задания ЛР (разделы 3,4).
4. Проверить в данном проекте примеры из методических указаний, выполнив их в отладчике в пошаговом режиме (раздел 4).
5. Написать программу заданий ЛР по варианту, выданному преподавателем и отладить ее (раздел 5).
6. Выполнить все контрольные задания (раздел 4)
7. **Продемонстрировать работу программы преподавателю в режиме отладчика по шагам и изменяемыми переменными.**
8. Подготовить отчет по ЛР по представленному шаблону (раздел 8).
9. Защитить ЛР с предоставлением отчета и ответами на контрольные вопросы (раздел 8,9).
10. Для продвинутых студентов выполнить задания для дополнительных (необязательных) требований и также отобразить их в отчете по ЛР (раздел 7).

3. 3. Основные понятия

3.1. 3.1 Линейные вычислительные процессы.

Линейные программы состоят из последовательности операторов ввода-вывода и операторов присваивания, которые выполняются в том порядке, в каком они записаны. Текст программы на СИ может быть разделен на несколько исходных файлов (в проекте), каждый из которых представляет собой текстовый файл, содержащий либо всю программу, либо ее часть. При компиляции исходной программы каждый из ее составляющих файлов компилируется отдельно, а затем связывается компоновщиком с другими файлами. Отдельные файлы можно

объединить в один посредством директивы препроцессора **include**. Иногда удобно в одном файле размещать переменные, а в других файлах использовать эти переменные путем их объявления. Каждая программа содержит главную функцию **main**. Если программа аргументов командной строки не содержит, то функцию **main** можно объявить без параметров.

3.2. 3.2 Пример простейшей программы на языке СИ.

Пример простейшей программы на языке СИ приведен ниже. В строках – комментариях, помеченных слешами («//») даны пояснения для каждой строки программы. Для выполнения этой программы ее необходимо ставить в главный модуль проекта.

```
// Заголовочный файл библиотеки ввода и вывода.(это комментарий)
#include <stdio.h>
// Название главной программы на СИ
void main(void)
{
    // Вызов функции вывода данных на экран дисплея
    printf("HELLO!!!\n");
    // Вызов функции ввода символа с клавиатуры
    getchar();
    //
}
```

Данная программа выводит на экран командной строки текст - "HELLO!!!", переводит строку и ожидает нажатия любой клавиши на клавиатуре. После этого она завершает работу.

3.3. 3.3 Программные проекты в системах программирования на СИ.

Для профессионального программирования в системах программирования создается проект, содержащий исходные модули (текстовые файлы) программы. Модули могут быть двух видов: программные (файлы имеют расширения ***.cpp** или ***.c**) и заголовочные (файлы имеют расширения ***.hpp** или ***.h**). При компиляции и компоновке проекта используются единые настройки для всех модулей (1). Компилируются только те модули, которые с предыдущей компиляции изменялись (2). Это позволяет экономить время на новое построение проекта (сборку -build). Важным свойством проектов является то, что настройки, сделанные один раз для этого проекта сохраняются для следующих запусков проекта(3). Цифрами в конце предложений помечены главные свойства и преимущества программных проектов. Кроме того, отмечу, что проекты могут быть разных типов: консольные проекты, проекты для **Windows** – приложений, **WEB** – проекты Интернет, и многие другие. Мы будем использовать консольные проекты для изучения основ программирования на языке СИ.

3.4. 3.4 Создание консольного программного проекта в СП на СИ.

Для создания консольного проекта необходимо:

- Запустить систему программирования VS 2005/8/10/12/17/19;
- В меню **“File”** выбрать пункт **“New”** и в подменю выбрать позицию **“Project...”**;
- В списке **“Project types”** выбрать **“Visual C++/Win32”**, а в списке **“Templates”** выбрать **“Win32 Console Application”**;
- В поле **“Name”** ввести: LAB1_OP_DD (где DD – номер варианта по журналу группы текущего семестра. Например, для студента группы УЦ5-31 с вариантом 5 – введем – LAB1_OP_5). Далее нажать **“OK”**;

- В новом окне мастера проектов нажать “**Next**”. Проверить настройки проекта: “**Application Type**” должно быть – “**Console Application**”, “**Additional option**” -> “**Empty Project**” должен быть включено. Остальные галочки должны быть выключены.
- Далее необходимо нажать кнопку “**Finish**”. Новый проект будет создан.
- Далее нужно добавить исходный модуль в проект: модуль LAB1_OP_DD (у нас это модуль LAB1_OP_5.CPP) добавляется в раздел “**Source Files**”. Нажмем правую кнопку на этом тексте, а затем: “**Add**” -> “**New Item ...**” -> “**Code**” -> “**C++ File**” -> Ввод поля “**Name**”;
- Далее нужно добавить заголовочный модуль в проект: LAB1_OP_5.H (у нас в примере LAB1_OP_5.H) - в раздел “**Header Files**” (Заголовочные файлы). Нажмем правую кнопку на этом тексте, а затем: “**Add**” -> “**New Item ...**” -> “**Code**” -> “**Header File**” -> Ввод поля “**Name**”;
- В файл LAB1_OP_5.CPP нужно занести информацию расположенную ниже:


```
#include "lab1_OP_5.h" // Поправить индекс группы и вариант
#include <process.h>
#include <stdio.h> // Подключение библиотеки ввода вывода
void main(void)
{
    // ...
}
```
- Файл LAB1_OP_5.H можно/нужно оставить пустым.
- Для контроля правильности создания пустого проекта, нажмем клавишу “F7” для проверки возможности создания программы (**build**) и “F5” для проверки ее выполнения (run/**debug**). Все перечисленные действия и операции должны быть выполнены без ошибок и предупреждений со стороны системы программирования СИ.

3.5. 3.5 Русификация проекта консольного ввода и вывода.

Для корректного отображения текстов на русском языке и его ввода в окне командной строки (после первого запуска программы) нужно сделать настройки шрифта этого окна. Переключаем шрифт в тип - Lucida Console. Выбираем настройки (после вывода консольного окна на экран, правой кнопкой вызываем системное меню): СВОЙСВА->ШРИФТ -> Lucida Console). После переключения шрифта, на запрос в отдельном окошке нужно выбрать режим – “Для всех окон с данным именем!”. Для правильной русификации окна консоли, кроме этого, в самом начале главной программы нужно переключить кодовую страницу для вывода:

```
system(" chcp 1251 > nul");
```

Для приостановки завершения программы в консольном окне в конце ее работы можно вызвать паузу следующим образом (например, в конце текста программы):

```
system(" PAUSE");
```

На экране появиться следующая строка (смотри ниже) и программа будет ожидать нажатия клавиши:

Для продолжения нажмите любую клавишу . . .

Примечание. Обратите внимание на то, что при другом способе локализации (setlocale(0,"rus")); не все в программе работает правильно. Вывод на консоль и ввод с консоли выполняется правильно, но после этого введенные в консольном окне данные (например, строка) имеют другую кодировку и выводятся неверно! Можете сами это проверить. Поэтому предпочтительно использовать предложенный выше способ с переключением кодовой страницы.

ОП ГУИМЦ 2024 ЛРН№1

Примечание. Если вы затрудняетесь выполнить заданный пункт ЛР, обратитесь к разделу “Основные понятия”, где приведены примеры для иллюстрации данного пункта.

Примечание. Все описания функций и общих данных нужно выполнять в заголовочном файле **LAB1_XDD.H** (у нас в примере **LAB1_15.H**). Программу нужно разместить в основном файле: **LAB1_XDD.CPP (LAB1_15.CPP)**.

Далее будет представлено несколько разделов, понятия которых также будут разъяснены на лекции, но они необходимы для выполнения первой ЛР.

3.6. 3.6 Программы и программирование

Программа – это упорядоченная совокупность операторов языка (S), которые определяют действия (шаги) для реализации поставленной задачи на основе разработанного алгоритма. Программа в самом общем виде предназначена для преобразования информации (данных), поэтому при программировании значительную роль играют данные и их структуры.

Схематично любая программа может быть представлена так:

< S1 , S2 , S3 , ..., Si ,... , Sk >

Здесь **Si** - это либо директива описания переменных, либо оператор вычисления значений, либо оператор управления, либо оператор вызова функций. Так как операторы и директивы записаны в определенном порядке, то они выполняются, последовательно. Специальные операторы (ветвления, циклов, вызова функций) могут изменить последовательность выполнения операторов. Правила записи операторов определяются конкретным языком программирования (точнее его синтаксисом), которые формально и точно определяют способы записи операторов и описаний данных. В нашем курсе мы используем язык C/C++, хотя многие понятия являются общими и для других языков программирования.

Программирование – это процесс создания программ для компьютера. Для этой работы применяются специальные программные комплексы – системы программирования. Системы программирования включают в себя много разных сервисных программ (например, компиляторов, отладчиков и т.д.), которые упрощают работу и делают ее более производительной. Кроме этого в системы программирования включаются специальные библиотеки, значительно упрощающие процесс написания сложных программ и их отладку. Наличие в библиотеках различных программ, готовых к использованию, и которые можно подключить во вновь разрабатываемые программы, делает процесс программирования более эффективным. Библиотечные программы называют подпрограммами или функциями.

Процесс программирования включает следующие этапы:

- Осмысливание задачи, которую нужно решить путем создания программы;
- Разработка алгоритма решения задачи;
- Выбор подходящего языка программирования для реализации программы;
- Разработка формализованных описаний алгоритма и логического проекта (блок-схемы и диаграммы классов, диаграммы объектов);

Написание программы на языке программирования и безошибочный ввод ее в компьютер;

Отладка программы, включающая все шаги (компиляции, редактирования связей, создание исполнимого модуля, пошагового исполнения программы с контролем промежуточных и окончательных результатов).

Оформление документации на программный продукт и предоставление ее заказчику.

Эти этапы могут быть простыми для простых задач и трудоемкими для сложных проектов, они могут занимать продолжительные периоды времени.

3.7. 3.7 Переменные

При написании (создании) программ, как было сказано выше, большое значение имеют данные, которые могут для разных целей представляться в разной форме. Форма (или формат) представления данных называется **типом** данных. Для работы с данными в программе им присваиваются специальные имена, которые иногда называются идентификаторами данных. Данные в программе могут быть постоянными и изменяемыми. Постоянные данные называются **константами** (см. ниже). Изменяемые данные называются **переменными**. Таким образом, переменной называется элемент программы, который имеет уникальное имя в программе и специальный тип. Программа работает с именами переменных, которые программист может задавать сам. Отметим, что переменным соответствуют области оперативной памяти компьютера, в которых запоминаются их текущие значения во время выполнения программы. Тип переменным необходим для указания операторам программы того, какие действия над конкретными переменными можно выполнять и сколько оперативной памяти нужно выделить для них размещения. В языке C++ используется простая форма для описания переменных:

<тип переменной> <имя/название/идентификатор переменной>;

В язык заложен набор стандартных типов переменных (int, long, char и т.д.) и специальные механизмы описания новых типов. Для описания новых типов переменных используются классы. Стандартные библиотеки языка также предлагают большой набор новых типов и операций над ними. С классами мы познакомимся в других ЛР. Переменные в программе описываются по следующему правилу:

<тип переменной> <имя переменной> = <значение для инициализации>;

Примеры описания простых переменных стандартных типов C++ показаны ниже:

```
// Простые переменные разных типов
int j = 5; // переменная целого типа
unsigned char c = 'A'; // переменная символьного типа
long l; // переменная целого типа длинная
float f = 5.5f; // переменная вещественного типа
double d = 10.00; // переменная вещественного типа длинная
bool b = true; // переменная логического типа (в базовом СИ нет такого типа)
string s; // переменная типа строка из библиотеки STL (в базовом СИ нет)
```

Подчеркну еще раз, что для переменных выделяется специальная область в оперативной памяти (ОП), в которой можно записывать и перезаписывать значения данных. Напомню, что оперативная память это специальное устройство компьютера, которое непосредственно связано с микропроцессором и служит для хранения программ и данных. Для доступа к переменным и командам используются адреса в оперативной памяти. В C++ можно явно использовать в программах адреса переменных. Для этого применяются специальные типы переменных: указатели и ссылки.

3.8. 3.8 Константы

Кроме переменных, в операторах и операциях можно использовать константы. Константы не могут изменяться и в большинстве случаев не хранятся в оперативной памяти. Они заменяются в исходном тексте программы. Константы записываются на основе правил, определенных в языке и бывают двух основных типов: числовые и символьные. Для группового описания констант используются перечисления (**enum**). Примеры использования констант показаны ниже.

```
int i = 5; // константа целого типа,
double d = 5.2; // вещественная константа, используемая для инициализации
d = d*5 + 11.7; // константы в выражении
string Str1 = "Пример строки 1 "; // строковая константа в " ... "
```

В языке C++ можно использовать также переменные константного типа. Для этого используется специальное ключевое слово **const**. Для таких переменных также выделяется оперативная память, но в программе их изменять нельзя. Константные переменные должны быть обязательно инициализированы. Примеры:

```
const int i = 3; // константа целого типа,
const double d = 3.3; // вещественная константа, используемая
// для инициализации
const float f = 5.5F; // вещественная константа float, нужна
// спецификация ("F") константы
```

3.9. 3.9 Операторы и составные операторы

Для работы с константами и переменными используются операторы программы (S_i). Операторы подразделяются на две группы: операторы изменения переменных и операторы управления. Основным оператором изменения переменных, является оператор присваивания, который имеет вид:

<левая часть выражения присваивания> = <правая часть выражения присваивания>;

В левой части оператора присваивания (в общем случае выражения) чаще записывается конкретная переменная, а в правой части выражение аналогичного типа. Например:

```
a = b + c; // a, b, c - переменные одного типа
Str3 = Str1 + Str2; // переменные Str1, Str2, Str3 имеют тип string (для C++)
```

Операторы управления рассмотрим ниже. В языках структурного программирования используется понятие составного оператора. Составной оператор – это любая совокупность операторов заключенная в специальные операторные скобки. В C++ операторными скобками являются фигурные скобки (“{”, “}”). В других языках программирования могут быть и другие операторные скобки (например, **begin** и **end**). В принципе, существуют языки, в которых не применяются операторные скобки и для объединения операторов используется специальное структурное текстовое представление программы (например, **OUTLINE** как в **WORD**). Составной оператор можно записать так:

{ S1 , S2 , S3 , ..., Si ,... , Sk } ;

Где **Si** - любые операторы и директивы описания языка C++.

Примеры составных операторов:

```
{ i = 5 ; c = (a>b) ? a : b; fun(a , 15); };
{
    if ( a > b)
    { int i =5; a = 0; b = 15 + i;}
    else
```

ОП ГУИМЦ 2024 ЛРН№1

```
{
    a = 15; b = 0; };
};
```

В данном примере использованы также операторы ветвления (if – else и условный оператор () ? :).

В языке программирования C/C++ состав стандартных операторов значительно ограничен. Он легко запоминается. Перечень этих операторов дан ниже (в СИ):

- Оператор присваивания “-”.
- Операторы ветвления (if – else).
- Операторы цикла (for, do, while).
- Оператор переключатель (switch).
- Оператор вызова функций и процедур .
- Оператор возврата из процедур (return).
- Вспомогательные операторы управления (break, continue, case)

В данной работе мы будем изучать оператор присваивания, который используется для вычисления значений переменных программы.

3.10. 3.10 Первая программа и русификация


Если создать пустой проект (см. выше – 3.4) – для примера - FIRST и добавить в него один иодуль **first.cpp** (файл исходного кода), а затем ввести в него текст программы, показанной ниже, то можно выполнить первую программе на языке C/СИ++. Для этого нужно ее откомпилировать (F7) и запустить ее в режиме отладки (F5).

```
#include <stdio.h>    // - 1
void main(void) // - 2
{ // Вывод текста
printf( "HELLO!, Привет!!!\n" ); // - 3
getchar(); // - 4
}
```

Первоначальные пояснения к тексту первой программы (Зеленым цветом выделены комментарии в тексте программы, номер комментария соответствует пояснению в перечислении, данному ниже). Красным цветом выделены фигурные операторные скобки ({,}) для обрамления тела главной программы.

1. Подключение библиотеки ввода/вывода (для использования функций **printf** и **getchar**).
2. Описание заголовка главной функции программы (**main**), такая необходима всегда.
3. Вызов вывода – **printf** для вывода текстовой строки ("HELLO!, Привет!!!"). В строке специально помещен текст в русской и латинской кодировке. Далее будет дано пояснение использования этой функции.
4. Вызов функции **getchar** для создания ожидания нажатия любой клавиши.

Полученный результат в окне консоли, после запуска программы, правда, в этом случае, будет не очень наглядным и понятным:



```
HELLO!, Привет!!!
```

Такое происходит потому, что не обеспечена русификация проекта и вывода на экран (слово "Привет" " печатается не в той кодировке – ("Ѕёштх€!!!!)).

Для русификации вывода в первой программе изменим ее так:

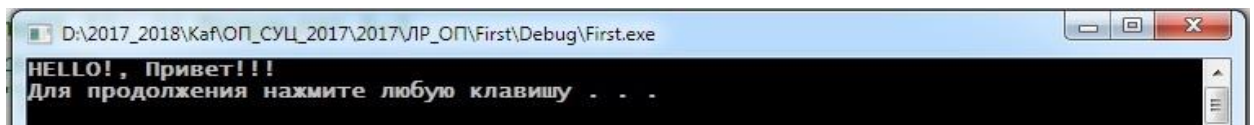
```
#include <stdio.h>
#include <process.h> // !!! - 1
void main(void)
{ // Вывод текста
// Руссификация ввода и вывода в консольном окне
system(" chcp 1251 > nul"); // !!! 2
printf( "HELLO!, Привет!!!\n" );
system(" PAUSE"); // 3// !!! 3
};
```

Красным в тексте помечены добавленные строки, которые обозначают следующее:

1. Добавлена вторая библиотека, для использования специальной функции **system**.
2. Вызывается функция **system** для переключения кодовой страницы консольного окна на русскую (1251).
3. Вызывается функция **system** для организации ожидания нажатой клавиши.

Кроме того, для корректного вывода, нужно настроить консольный ввод и вывод (см. выше 3.5- русификация проекта). Повторю здесь еще раз настройки шрифта:

Для отображения текстов на русском языке и его ввода в окне командной строки (после первого запуска программы!) нужно сделать настройки шрифта этого окна (правой кнопкой мыши вызываем у системного меню окна "-" левый верх окна консоли). Обязательно переключаем шрифт в тип - **Lucida Console**. Выбираем настройки (после вывода консольного окна на экран, правой кнопкой вызываем системное меню): СВОЙСВА->ШРИФТ -> Lucida Console). В результате получим:



После нажатия любой клавиши консольное окно закроется. Действия по настройке окна нужно делать только при первом запуске на выполнение для данного проекта!

3.11. 3.11 Подключение функций математической библиотеки.

Для использования математической библиотеки нужно подключить заголовочный файл

```
#include <math.h>
```

Он содержит перечень прототипов функций и констант (например π), которые используются в формулах. Перечень функций приведен в приложении (11.1), кроме этого можно посмотреть содержимое заголовочного файла(math.h).

Для подключения констант нужно использовать следующую настройку:

```
#define _USE_MATH_DEFINES
```

Тогда в программе можно использовать математические константы (например π) - M_PI.

Перечень допустимых констант дан в приложении (11.1)

3.12. 3.12 Программа вычисления синуса.

С учетом пояснений, сделанных выше, в созданном проекте можно создать программу (см. ниже) для вычисления значения функции синус (по формуле, представленной ниже):

$$F = 2 * \sin(\pi / 2):$$

Пример фрагмента программы для вычисления синуса для разных значений аргумента (с подключенной математической библиотекой и русификацией):

```
#define _USE_MATH_DEFINES
// Подключение библиотеки математических функций
#include <math.h>
#include <stdio.h>
#include <process.h>
void main(void)
{
    // // п. 3.12
    system(" chcp 1251 > nul");
    double F;
    double X = M_PI/2; // X =  $\pi / 2$  – начальное значение
    //////////
    F = 2*sin(X); // F = 2.0 2*sin( $\pi/2$ )
    //////////
    printf("1. F = %7.2f для X =%5.2f \n" , F , X );
    F = 2*sin(X + M_PI/2 ); // F  $\approx$  0.0 x =  $\pi$ 
    printf("2. F = %7.2f для X =%5.2f \n" , F , X );
    F = 2*sin(2*X ); // F  $\approx$  0.0 sin( $\pi$ )
    printf("3. F = %7.2f для X =%5.2f \n" , F , X );
    F = 2*sin(M_PI/2 ); // F  $\approx$  0.0 x =  $\pi$ 
    printf("4. F = %7.2f для X =%5.2f \n" , F , X );
    F = 2*sin(0.0 ); // F = 2.0 2*sin(0)
    printf("5. F = %7.2f для X =%5.2f \n" , F , X );
    F = 2*sin(1.57079632679489661923 ); // F = 2.0 sin( $\pi$ )
    printf("6. F = %7.2f для X =%5.2f \n" , F , X );
    system(" PAUSE"); }
```

Результаты работы программы можно посмотреть в отладчике в окне локальных данных или вывести на экран консольного окна функцией **printf** спецификацией (%7.2f). Смотрите ниже примерв форматированного ввода и вывод.

```
printf("F = %7.2f для X =%5.2f \n" , F , X);
```

После вычисления получим результат ($X = \pi/2$):

```
F = 2.00 для X = 1.57
```

Особенности ввода вывода чисел и текстов мы рассмотрим ниже.

3.13. 3.13 Форматированный ввод-вывод.

Рассмотрим простейшие возможности ввода/вывода в стандартной библиотеке.

Для этого используются функции **printf** (вывод на консоль – окно командной строки) и функция **scanf** (ввод с клавиатуры – в режиме командной строки). Для подключения библиотеки ввода/вывода используется заголовочный файл **<stdio.h>**:

```
#include <stdio.h>
```

```
#include <process.h>
```

Рассмотрим главные функции.

3.14. 3.14 Вывод данных – функция printf.

Функция **printf** (вывод по формату), является основной функцией вывода на консоль в языке СИ (в языке СИ++ предусмотрены специальные классы для ввода и вывода с консоли, но эти возможности будут рассматриваться потом). Первый аргумент функции – строка форматирования, она задается в двойных кавычках. Эта строка выводит текст и определяет формат вывода переменных. Пример вывода текста без переменных:

```
printf("Пример сообщения\n"); // выведет на печать только текст и переведет строку (\n):
```

Пример сообщения

А функция выводит текст и значение одной переменной целого типа (**i = 5**):

```
int i = 5;
printf("Длина равна = %4d см. \n", i);
printf("Длина равна = %4d см", i);
// при i = 1 выведет:
```

Длина равна = 5 см

Таким образом, запись **"%4d"** означает: печатать переменную заданную вторым аргументом величину в десятичной (**d** – **тип вывода**) форме в следующие 4 позиции. Способ записи формата следующий:

%[<общий размер>][.<точность>]%[.<Тип вывода>]<тип вывода>

Кроме десятичного спецификатора формата вывода могут быть применяться и другие:

d -преобразование в десятичный формат;

o -преобразование в восьмеричный формат;

x-преобразование в шестнадцатеричный формат;

n-используется для запоминания текущей позиции в строке вывода;

f,g,e – печать действительных типов с порядком (**e**) и без.

p – выводится в формате адреса (указатель)

s-печать в формате строки (передается указатель);

c-печать одного символа.

Точность указывает для дробной части число знаков после запятой.

Пусть **pS**-указатель на строку с содержанием **"Первое значение"**, а переменная **x**- содержит вещественное число равное **5.8134f**, тогда оператор :

```
float x = 5.8134f;
char pS[] = "Первое значение";
printf("%s x = %6.3f \n", pS , x );
```

выведет на печать:

Первое значение x = 5.813

'\n' - в строке форматирования означает перевод строки. Под число с плавающей точкой отведено всего **6** позиций, три из которых выделено под дробную часть (точность). Это справедливо для всех типов данных. Тип вывода **f** используется для вывода данных типа **float** (число с плавающей точкой). Для вывода числа типа **double** нужно указать формат для вывода - **%lf** (спецификатор **l** для long/double). Форматная строка в функции **printf** может содержать также специальные управляющие символы:

\n- перевод на новую строку;

ОП ГУИМЦ 2024 ЛРН№1

\f-новая страница;

\t-табуляция;

\b-стереть предыдущий символ и т.д.

Еще один пример печати одного символа(**symb**), целого числа (**numb**) и строки(**Str**).

```
char symb='Ж';
int numb = 33;
char Str[25] = "Строка текста";
printf ("%10.3s %c %2d" , Str, symb, numb);
system(" PAUSE");
```

Получим:

Первое значение x = 5.813

Стр Ж 33

Для продолжения нажмите любую клавишу . . .

Под строку отведено минимально 10 символов, но печатается только 3 (спецификация - %10.3s).

3.15. 3.14 Ввод данных – функция scanf.

В языке СИ для ввода данных используются различные функции, в том числе **getchar()** и **scanf()**. Простой пример ввода и вывода:

```
char c;
c=getchar();
printf("Введите символ \"c\":\n");
printf("Символ = %hc !!!\n", c);
```

После ввода "5" и нажатия клавиши Enter, Получим на экране.

Введите символ "с":

5

Символ с = 5 !!!

Здесь **c** - переменная типа **char**, ей будет присвоено значение (5), введенное с клавиатуры. Спецификатор типа **h** – используется для вывода типа **short int** переменных.

Ввод других типов данных с клавиатуры (не только одиночных символов) выполняется также с помощью функции **scanf**. В этом случае вводимые параметры/переменные задаются всегда с помощью указателей(адресов переменных). Пусть переменные имеются три переменные: **symb** , **numb** и **Str**:

```
char symb='A';
int numb = 11;
char Str[25] = "Строка текста"; // Имя массива является указателем.
```

Тогда можно выполнить ввод данных:

```
printf ("Введите строку, символ и число [через пробел]:");
scanf("%s %c %d", Str, &symb, &numb);
printf ("%10.5s %c %2d\n" , Str, symb, numb);
```

Введите, символ и число [через пробел]:

Пример_6_77_

После ввода слова "Пример" символа "6" и числа "77"

Получим:

Приме_6_77

Все аргументы представляют собой указатели (и также **Str** - имя символьного массива строки это тоже указатель). При вводе функция **scanf** рассматривает пробелы как разделители вводимых данных. **Красным** цветом выделена символьная переменная (**Str**) и ее значения при вводе и выводе данных.

Примечание: Для дополнительных сведений о работе функций можно воспользоваться также: литературой по курсу, лекциями по курсу, MSDN и другими источниками информации.

Перечень доступных функций ввода вывода вы найдете в библиотеке **<stdio.h>**.

3.16. 3.16 Отладка программ.

При разработке программ важную роль играет отладчик, который встроен в систему программирования. В режиме отладки можно проверить работоспособность программы и выполнить поиск ошибок самого разного характера. Отладчик позволяет проследить ход (по шагам) выполнения программы и одновременно получить текущие значения всех переменных и объектов программы, что позволяет установить моменты времени (и операторы), в которые происходит ошибка и предпринять меры ее устранения. В целом, отладчик позволяет выполнить следующие действия:

- Запустить программу в режиме отладки без трассировки по шагам (**F5**);
- Выполнить программу по шагам (**F10**);
- Выполнить программу по шагам с обращениями к вложенным функциям(**F11**);
- Установить точку останова (**BreakPoint – F9**);
- Выполнить программу до первой точки останова (**F5**);
- Просмотреть любые данные в режиме отладки в специальном окне (**locals**);
- Просмотреть любые данные в режиме отладки при помощи мышки;
- Изменить любые данные в режиме отладки в специальном окне (**locals** и **Watch**);
- Установить просмотр переменных в специальном окне (**Watch**);
- Просмотреть последовательность и вложенность вызова функций.

При выполнении всех лабораторных курса студенты должны активно использовать отладчик **VS**, знать его возможности и отвечать на контрольные вопросы, связанные с отладкой и тестированием программ.

Примечание. Подробное описание материала и понятий вы можете найти в литературе [1 - 6] или справочной системе MS VS. Кроме того, не пропускайте лекции по курсу. Не рекомендую безоговорочно верить материалам из сети Интернет (например, в Википедии), так как там в некоторых статьях есть ошибки!

3.17. 3.17 Пример программы линейного вычисления с вводом и выводом данных.

Пусть необходимо создать программу для вычисления значения переменной **F** (переменная с плавающей точкой) на основе следующей формулы:

$$F = 2ab \sin(l + 0.5\pi)$$

Параметры вычисления **a** и **b** являются целыми переменными (тип - **int**), а переменная **l** является переменной с плавающей точкой (вещественной переменной). Параметры вычисления должны быть введены с клавиатуры (с консоли), а результат должен быть выведен в окно командной строки. Тогда можно создать программу, приведенную ниже и выполнить необходимые вычисления. Пояснения в тексте программы даны в виде комментариев(//).

```

#define _USE_MATH_DEFINES
// Подключение библиотеки математических функций
#include <math.h>
#include <stdio.h>
#include <process.h>
// описание константы для числа пи
#define PI 3.14f
// Начало программы – точка входа в программу
void main(void)
{
// Руссификация ввода и вывода в консольном окне
system(" chcp 1251 > nul");
// Описание исходных данных и вычисляемых переменных
int a , b; // переменные целого типа
double F , l ; // переменные вещественного типа
// Ввод данных a
printf("Введите a: ");
scanf_s("%d", &a);
// Ввод данных b
printf("Введите b: ");
scanf_s("%d", &b);
// Ввод данных l
printf("Введите l: ");
scanf_s("%lf", &l);
// Вычисление по формуле
F = (double) (2*a*b*sin(l+0.5f*M_PI));
// Вывод результата работы программы
printf("\nF = %7.2lf для a = %d b = %d l = %5.2lf \n" , F , a , b , l);
// Ожидание завершения работы программы
system(" PAUSE");
};

```

Блок-схема линейной программы вычисления формулы приведена ниже (блок ввода данных №2 можно детализировать):



ОП ГУИМЦ 2024 ЛРН№1

Результаты работы такой программы после ввода необходимых переменных выглядят так (проверьте ее работу на аналогичных данных):

Введите a: 2

Введите b: 3

Введите l: 10

F = -10.07 для a = 2 b = 3 l = 10.00

Для продолжения нажмите любую клавишу . . .

Нужно создать пустой проект в MS VS, как описано выше (раздел 3.4 и 3.5), скопировать через буфер обмена в него текст данного примера прямо из текста данного документа, отладить его (Раздел 3.11) и выполнить пошагово.

3.18. 3.18. Болванка для главного модуля с математической библиотекой

Нужно создать пустой проект в MS VS, как описано выше (раздел 3.4 и 3.5), скопировать через буфер обмена в него текст данного примера прямо из текста данного документа, отладить его (Раздел 3.11), русифицировать (п.3.5) консольное окно и выполнить пошагово. Выбрать один из способов ветвления в программе (переходы или переключатель). Ненужное можно удалить.

```
#define _USE_MATH_DEFINES

#pragma warning(disable : 4996)

// Подключение библиотеки математических функций
#include <math.h>
#include <stdio.h>
#include <process.h>
void main(void)
{
    // // п.
    system(" chcp 1251 > nul");
    // Здесь расположить тексты примеров и операторов вычислений из МУ
    // Пример печати
    printf("Главный модуль простого примера с математикой!\n");
    system(" PAUSE");
}
```

3.19. 3.19. Болванка для демонстрации ЛР или работы на семинаре

Нужно создать пустой проект в MS VS, как описано выше (раздел 3.4 и 3.5), скопировать через буфер обмена в него текст данного примера прямо из текста данного документа, отладить его (Раздел 3.11), русифицировать консольное окно и выполнить пошагово.

```
#define _USE_MATH_DEFINES
// Подключение библиотеки математических функций
#include <math.h>
#include <stdio.h>
#include <process.h>
void main(void)
{
```

ОП ГУИМЦ 2024 ЛР№1

```

// // п.
system(" chcp 1251 > nul");
// Здесь расположить тексты примеров из МУ
////////////////////////////////////
// 1-й СПОСОБ для демонстрации и занятий
// для использования в работе может быть выбран первый или второй способ
// (они взаимоисключающие!) Управление демонстрацией производится установкой breakpointa

// для этого переход на метку конца нужно закомментировать так : //goto METBEGIN;

goto METBEGIN;

// НАЧАЛО

// 1
METBEGIN;; // Метку перетащить к началу составного оператора проверяемого пункта или задания или
поставить точку останова
{
    // п 5.6
    int i =5;
    //
    printf("1. П. 5.6\n");

goto METEND;}; ;
// 2

                { // п 5.7
printf("2. П. 5.7\n");
goto METEND;}; ;
// ...
// ...
// КОНЕЦ
METEND;; // Метка конца проверки
// 2-й СПОСОБ для демонстрации и занятий (сначала выводится меню, а затем используется
переключатель)

char SW;
// Вывод меню
printf("1. П. 5.6\n");
printf("2. П. 5.7\n");
// ...
// выбрать нужный пункт меню
printf("Выберете пункт меню:\n");
SW = getchar(); // Ввод нажатой клавиши (номера пункта меню)
// Переключатель

switch ( SW)
{
    case '1':
    {
printf("Выбор 1!\n");
// проверяемая программа
// п.5.6
break;

    }
    case '2':
    {
printf("Выбор 2!\n");

```

ОП ГУИМЦ 2024 ЛР№1

```
//проверяемая программа
//п.5.7
break;

    case '3':
    {
printf("Выбор 3!\n");
break;
break;
    }

    default:
    {
printf("Выбор по умолчанию!!\n");
    }

};

system(" PAUSE");
}
```

3.20. 3.20. Болванка с меню, выходом и возвратом в меню

В данном разделе приводиться текст "болванки" с меню, возможностью выхода из программы и возврата в меню"

```
#define _USE_MATH_DEFINES
// Подключение библиотеки математических функций
#include <math.h>
#include <stdio.h>
#include <process.h>
void main(void)
{
// // п.
system(" chcp 1251 > nul");
// Здесь расположить тексты примеров из МУ
////////////////////
// для использования может быть выбран один или второй способ (взаимоисключающие!)
// 2-й СПОСОБ для демонстрации и занятий

char SW;

MENU::;
// Вывод меню
system (" CLS");
MENU2::;
printf("1. П. 5.6...\n");
printf("2. П. 5.7...\n");
printf("3. П. 5.8...\n");
// ....
printf("е. Выход ...\n");
printf("0. Меню ...\n");

// ...
// выбрать нужный пункт меню
printf("Выберете пункт меню:\n");
SW = getchar(); // Ввод нажатой клавиши
// Переключатель

switch ( SW)
```

```

{
case '1':
{
printf("Выбор 1!\n");
// проверяемая программа - после этого пункта завершение проверки и программы !!
// п.5.6 ...
break;
}
case '2':
{
printf("Выбор 2!\n");
//проверяемая программа - после этого пункта выход в меню
//п.5.7 ...
// system (" CLS");
SW = getchar(); // Сброс ENTER
goto MENU;
break;
}
case '3':
{
printf("Выбор 3!\n");
//проверяемая программа - пункта выход в меню с сохранением результата
//п.5.8 ...

SW = getchar(); // Сброс ENTER
goto MENU2;
break;
}
case '0':
{
printf("menu!\n");
// После этого пункта очистка и возврат в меню
SW = getchar(); // Сброс ENTER
goto MENU;
break;
break;
}
case 'e':
{
printf("Выбор e !\n");
// После выбора этого пункта выход из программы
exit (0);
}
default:
{
printf("Выбор по умолчанию!!\n");
}

};

system(" PAUSE");
}

```

3.21. 3.21. Болванка с меню, выходом и возвратом в меню для ДЗ и выходом

```

#define _USE_MATH_DEFINES
// Подключение библиотеки математических функций
#include <math.h>

```

```

#include <stdio.h>
#include <process.h>
#include <process.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#include <time.h>
#include <io.h>
/// ВРЕМЕННО ЗАКОМЕНТИРОВАНО #include "header.h"
extern int Counter;
void main(void)
{
    // // п.
    system(" chcp 1251 > nul");
    // Здесь расположить тексты примеров из МУ
    //////////////////////////////////////
    // 2-й СПОСОБ для демонстрации и занятий

    char SW;

    MENU::;
    // Вывод меню
    system (" CLS");
    MENU2::;
    printf("1. П. ТЗ 5.1.1 ...\\n");
    printf("2. П. ТЗ 5.1.3 ...\\n");
    printf("3. П. ТЗ 5.1.3 ...\\n");
    printf("4....\\n");
    printf("*. П. ТЗ 5.X.X ...\\n");
    // ....
    printf("в/у/е. Выход ...\\n");
    printf("0. Меню ...\\n");

    // ...
    // выбрать нужный пункт меню
    printf("Выберете пункт меню:\\n");
    SW = getchar(); // Ввод нажатой клавиши
    // Переключатель

    switch ( SW)
    {
        case '1':
        {
            printf("Выбор 1!\\n");
            // проверяемая программа - после этого пункта завершение проверки и программы !!
            // п.5.6 ...
            break;
        }

        case '2':
        {
            printf("Выбор 2!\\n");
            //проверяемая программа - после этого пункта возврат в меню программы с очисткой
            //п.5.7 ...
            // system (" CLS");
            SW = getchar(); // Сброс ENTER
            goto MENU;
            break;
        }
    }
}

```

ОП ГУИМЦ 2024 ЛР№1

```

        case '3':
        {
printf("Выбор 3!\n");
//проверяемая программа - пункта выход в меню с сохранением результата
//п.5.8 ...

SW = getchar(); // Сброс ENTER
goto MENU2;
break;
        }

//////////
/// Шаблоны для case
/*
    Шаблон без очистки результата !!!!! убрать комментарии и заменить *
    case '*':
    {
printf("Выбор *!\n");
//проверяемая программа - пункта выход в меню без сохранения результата
//п.ТЗ ...

SW = getchar(); // Сброс ENTER
goto MENU;
break;
    }

    */
    /*
    Шаблон с очисткой результата !!!!! убрать комментарии и заменить *
    case '*':
    {
printf("Выбор *!\n");
//проверяемая программа - пункта выход в меню с сохранением результата
//п.ТЗ ...

SW = getchar(); // Сброс ENTER
goto MENU2;
break;
    }

    */
    //убрать комментарии

//////////
case '0':
{
printf("menu!\n");
// После этого пункта очистка и возврат в меню
SW = getchar(); // Сброс ENTER
goto MENU;
break;
break;
}

case 'y':
case 'e':
case 'b':
{
printf("Выбор в !\n");
// После этого пункта выход из программы
exit (0);

```

ОП ГУИМЦ 2024 ЛР№1

```

    }
default:
    {
printf("Выбор по умолчанию!!\n");
    }

};

system(" PAUSE");
}

```

При включении в проект своего заголовочного файла "header.h" нужно его создать, подключить в проект и убрать слеш (//) в его временном отключении в начале этого фрагмента.

4. 4. Контрольные задания ЛР №1.

4.1.4.1 Изучить теоретическую часть ЛР

Раздел 3 данных МУ. Это желательно сделать дома до начала ЛР или в самом начале ЛР.

4.2.4.2 Изучить создание проектов и создать первый проект (First).

Раздел 3.4 данных МУ. В проект подключается один модуль (**first.cpp**)

В этот модуль, который сначала пуст, скопировать из данных МУ пример простейшей программы (раздел 3.2), включая фигурные скобки. Перестроить проект, добившись отсутствия ошибок, если они возникли.

4.3.4.3 Выполнить русификацию проекта.

Раздел 3.5 данных МУ. Перестроить проект, добившись отсутствия ошибок, если они возникли.

4.4.4.4 Подготовить и выполнить первую простейшую программу по шагам.

Раздел 3.2 данных МУ.

4.5.4.5 Изучить и выполнить подключение математической библиотеки в проект и вычислить выражение, подключить константу Пи (π) и вычислить синус для выражения.

Раздел 3.12 данных МУ. Выполнить фрагмент программы вычисления синуса (3.12) для чего скопировать в текст ранее созданных проектов, представленных в данных МУ текст.

Раздел 3.12 данных МУ.

4.6.4.6 Изучить и проверить работу функций ввода и вывода.

Раздел 3.13 данных МУ.

4.7.4.7 Подготовить и выполнить в режиме отладки линейную программу с вводом, расчетом и выводом.

Раздел 3.15 данных МУ. Для этого скопировать текст программы (через буфер обмена) из МУ (Раздел 3.1) в модуль first.cpp.

Выполнить эту программу в пошаговом режиме (**F5** и **F10**) с просмотром всех переменных программы и результатов в консольном окне. При появлении ошибок или предупреждений исправить исходный текст и заново перестроить проект (**F7**). Ввести значения при запросе ввода: $a=1$, $b=1$, $l=1$.

Проверить полученный результат в консольном окне: **F=1.08**

4.8.4.8 Подготовить и выполнить программу по своему варианту.

Задание и его **варианты** для студентов заданий даны в Разделе 5 данных МУ (номер варианта берем из журнала группы). Подобрать такие вводимые значения (a, b, l), чтобы результат получался осмысленным и проверяемым.

Создать новый проект (LR1_31 или LR1_32, по индексу группы: ИУ5Ц-31Б или ИУ5Ц-32Б. Учесть номер по списку журнала группы. Написать программу на языке СИ для вычисления значения выражения по формуле варианта (см. ниже). Значения переменных **a**, **b** (целые переменные) и **l** (переменная с плавающей точкой) ввести с клавиатуры. При использовании математических функций необходимо использовать математическую. Для ввода - вывода использовать библиотеку ввода и вывода `#include <stdio.h>`. Вывести на консоль значения функции **F** и для разных входных величин (**a**, **b**, **l**). Завершить программу ожиданием нажатия любой клавиши ("**Pause**").

Нарисовать блок-схему программы.

4.9.4.9 Подготовить и выполнить программу с дополнительными требованиями

(раздел 6). Выполнить программу по шагам с точками останова. Табличко можно оформить с заголовком.

4.10.4.10 Оформить отчет по ЛР по представленному шаблону (в архиве с МУ ДР №1 на сайте) и с указанными рекомендациями (там же).

5. 5. Контролируемые требования ЛР №1.

- Создание и русификация проекта. - п.п 4.2/3.
- Выполнение простого примера. - п.п 4.4
- Подключение математической библиотеки функций и констант, - п.п 4.5
- Изучить и опробовать функции библиотеки ввода/вывода - п.п 4.6
- Выполнить простую программу с вводом, вычислением и выводом результата - п.п 4.7.
- Разработать и выполнить программу по своему варианту п.п 4.8 (см. таблицу вариантов) в том числе и в пошаговом режиме, точками останова и просмотром переменных.
- Продемонстрировать работающую программу ЛР по варианту. - п.п 4.8
- Подготовить отчет по шаблону и примеру и с учетом Требований
- с обязательной блок-схемой. п.п - 4.10

ОП ГУИМЦ 2024 ЛР№1

- Разработать программу с дополнительными требованиями (изменение переменной, цикл вывода данных в таблице). - п.п 4.9
- Ответы на все контрольные вопросы и умение выборочно на них отвечать.
- Умение выполнять все примеры из теоретической части ЛР п.п 4.1. МУ. (выборочно!)

6. 6. Варианты заданий для п.п.4.8 Задания ЛР №1.

Варианты заданий приведены ниже. Номер варианта должен соответствовать номеру студента в групповом журнале.

Вариант №	Формула для вычисления (а и b – целые –int , l – вещественная)
1.	1. $F = \frac{2(1-\pi)}{b^2 + l^2 - 1} \cos(0.5 * a)$
2.	2. $F = \frac{2(1-\pi)}{b^2 + l^2 - 1} + \frac{2\pi}{b^2 + l^2 - 1} \cos(0.1 * a)$
3.	3. $F = \frac{2(1-\pi)}{b^2 + l^2 - 1} + \frac{2\pi \cos(2 * a)}{b^2 + l^2 - 1}$
4.	4. $F = \frac{2(1-\pi)}{b^2 + l^2 - 1} + \frac{2\pi}{b^2 + l^2 - 1} \operatorname{tg}(3 * a)$
5.	5. $F = \frac{2(1-\pi)}{b^2 + l^2 - 1} + \frac{2\pi}{a^2 + l^2 - 1} \sin(3.0)$
6.	6. $F = \frac{a}{b} + \frac{2\pi}{b} \sin(0.4 * l)$
7.	7. $F = \frac{2(1-\pi) \sin(1)}{b^2 + l^2 - 1} + \frac{2\pi}{a^2 + l^2 - 1}$

Вариант №	Формула для вычисления (a и b – целые –int, l – вещественная)
8.	8. $F = \frac{2(1-\pi)}{b^2 + l^2 - 1} + \frac{2\pi}{b^2 + l^2 - 1} \operatorname{tg}(2)$
9.	9. $F = \frac{2(1-\pi) \sin(2 * a)}{b^2 + l^2 - 1} + \frac{2\pi}{b^2 + l^2 - 1}$
10.	10. $F = \frac{a}{b} \pi + \frac{2\pi}{b} \cos(0.3 * l * \pi)$

6.1.5.1 Демонстрация и защита.**6.2.5.2 Продemonстрировать работу пунктов ЛР преподавателю (демонстрация отмечается в журнале ЛР).**

Разделы 4.1 – 4.8 данных МУ.

6.3.5.3 Подготовить и защитить отчет по ЛР.

Раздел 7 данных МУ.

6.4.5.4 Прочитать и подготовить ответы на контрольные вопросы.

Раздел 8 данных МУ.

6.5.5.5 Выполнить программу с дополнительными требованиями.

Для продвинутых студентов. См. следующий раздел данных МУ.

7.7. Дополнительные требования для студентов СУЦ (д.т.).

Организовать цикл вывода значений функции для аргумента, который меняется с определенным шагом. Число выводимых значений не более 10-ти и задается при вводе исходных данных (int n), число n может быть задано преподавателем.

ОП ГУИМЦ 2024 ЛРН№1

Для вычислений использовать формулу **своего** варианта (см. задание выше). В цикле меняются значения **одного** из аргументов формулы: **a** , **b** или **l**. Целочисленные значения (a,b) меняются с шагом $h = 1$, а действительные аргументы (**l**) с шагом $h = \pi/2$. Изменяемую переменную в цикле (**a** , **b** или **l**.) задает преподаватель.

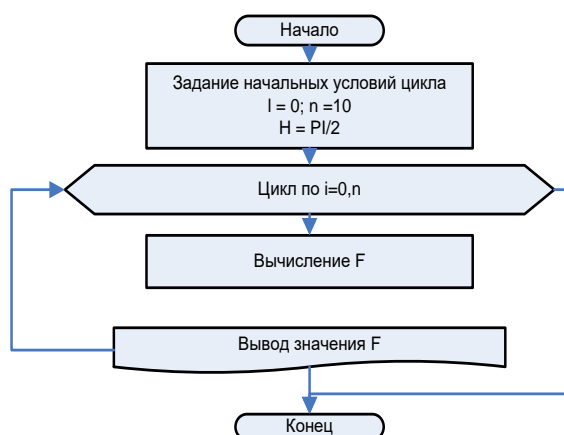
Создать вывод в цикле нескольких значений функции F для разных аргументов l . Для этого нужно организовать цикл по изменяемой переменной в виде таблицы обрамленной звездочками ("*") или точками(".") задает преподаватель.

$$F = 2absin(l + 0.5\pi)$$

1. Для нашего примера (см. выше) выбран изменяемый параметр **l** ($l = l + \pi/2$).
2. Пример фрагмента (цикла) программы приведен ниже:

```
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <process.h>
void main(void) {
// Руссификация ввода и вывода в консольном окне
system("chcp 1251 > nul");
// НУ Цикла
int a = 1 , b = 2; // переменные целого типа
float F ; // переменные вещественного типа
float h = (float)0.5f*M_PI;
int n = 10;
float l = 0.0f;
for (int i = 0 ; i < n ; i++)
{
F = (float) 2*a*b*sin(l+0.5f*M_PI);
printf("\n %6.2d. F = %7.2f для a = %d b = %d l = %5.2f " ,i+1 , F , a , b , l);
l += h ;
};
//
printf("\n ");
system(" PAUSE");
printf("\n ");}
```

Блок – схема первой циклической программы приведена ниже:



3. Пример вывода таблицы значений со звездочками.

```

*****
* № *      F *   a *   b *   l   *
*****
* 1. *    4.00 *   1 *   2 *   0.00 *
* 2. *    0.01 *   1 *   2 *   1.57 *
* 3. *   -4.00 *   1 *   2 *   3.14 *
* 4. *   -0.01 *   1 *   2 *   4.71 *
* 5. *    4.00 *   1 *   2 *   6.28 *
* 6. *    0.02 *   1 *   2 *   7.85 *
* 7. *   -4.00 *   1 *   2 *   9.42 *
* 8. *   -0.03 *   1 *   2 *  10.99 *
* 9. *    4.00 *   1 *   2 *  12.56 *
* 10. *   0.03 *   1 *   2 *  14.13 *
*****

```

Или с точками:

```

.....
. № .      F .   a .   b .   l   .
.....
. 1. .    4.00 .   1 .   2 .   0.00 .
. 2. .    0.01 .   1 .   2 .   1.57 .
. 3. .   -4.00 .   1 .   2 .   3.14 .
. 4. .   -0.01 .   1 .   2 .   4.71 .
. 5. .    4.00 .   1 .   2 .   6.28 .
. 6. .    0.02 .   1 .   2 .   7.85 .
. 7. .   -4.00 .   1 .   2 .   9.42 .
. 8. .   -0.03 .   1 .   2 .  10.99 .
. 9. .    4.00 .   1 .   2 .  12.56 .
. 10. .   0.03 .   1 .   2 .  14.13 .
.....

```

4. Использовать для организации цикла один из операторов языка СИ – **while** или **for** или **do**. Задаёт преподаватель.
5. Построить блок-схему своей программы с циклами в среде MS VISIO.

Примечание: Выполнение программы с дополнительными требованиями отображается в журнале, в отчете по ЛР и учитывается при подведении результатов работы в семестре и на экзамене.

8. 8. Демонстрация, защита ЛР и отчет по ЛР.

После выполнения всех необходимых шагов по ЛР, работающую программу нужно **продемонстрировать** преподавателю, проводящему ЛР, о чем он в журнале делает **отметку**. Далее студент на основе шаблона и примера оформляет **отчет** по ЛР и показывает его преподавателю (вместе с МУ по ЛР на сайте также размещены: шаблон отчета и пример отчета). После оформления отчета, часть которого может быть представлен преподавателю в электронном виде, выполняется защита ЛР. Студент должен дать ответы на любой вопрос по отчету и программе, а также уметь без подготовки ответить и на контрольные вопросы по ЛР

ОП ГУИМЦ 2024 ЛР№1

приведенные ниже. ЛР считается полностью **зачтенной**, если выполнены все перечисленные требования и действия: **демонстрация, отчет и защита ЛР**. О этом делается **отметка** ф журнале ЛР. Для допуска к экзамену нужно сдать и защитить все ЛР в срок.

9. 9. Срок демонстрации и защиты.

Срок демонстрации программы – 2 неделя семестра.

Срок защиты программы с отчетом – 3 неделя семестра.

10. 10. Контрольные вопросы по ЛР.

1. Что такое проект в VS 2005/8/10/12?
2. Для чего нужны проекты и в чем их преимущество (три)?
3. Как создать программный проект?
4. Дайте определение понятия программа.
5. Дайте определение понятия переменная.
6. Дайте определение понятия константа.
7. Дайте определение понятия составного оператора.
8. Как в программе можно выполнить ввод данных с клавиатуры?
9. Как в программе можно выполнить вывод данных в консольное окно?
10. Что нужно сделать для подключения математической библиотеки?
11. Что нужно сделать для подключения библиотеки ввода и вывода?
12. Что нужно сделать для русификации консольного окна?
13. Какие вы знаете программные проекты?
14. Как можно русифицировать консольный проект?
15. Как в программе можно задать число пи?
16. Как подключить математическую библиотеку?
17. Как подключить библиотеку ввода и вывода?
18. Какие функции для ввода и вывода вы знаете?
19. Что такое отладка программ, и какие действия можно предпринять с помощью отладчика?
20. Как организовать паузу в работе программы?
21. Какие режимы отладки программы вы знаете?
22. Какие можно приостановиться в определенном месте программы?
23. Какие фазы выполняются для создания программы?
24. Что такое сборка программы и как ее сделать?
25. Внесите изменения в вашу формулу и пересоздайте исполнимый модуль заново. Покажите результат преподавателю.
26. Как организовать цикл вывода значений функции для разных аргументов (д.т.)?
27. Какие операторы для организации циклов в СИ вы знаете (д.т.)?

11. 11. Литература.**Основная литература**

1. Список литературы, доступные книги и необходимые пособия для ЛР ОП размещены на сайте www.sergebolshakov.ru на страничке “2-й к СУЦ”. Пароль для доступа можно взять у преподавателя или старосты группы.

2. Керниган Б., Ритчи Д. К Язык программирования С, 2/3-е издание: Пер. с англ. – М. : Издательский дом “Вильямс”, 2009. – 304с.: ил. – Пар. Тит. англ.
3. Касюк, С.Т. Курс программирования на языке Си: конспект лекций/С.Т. Касюк. — Челябинск: Издательский центр ЮУрГУ, 2010. — 175 с.
4. MSDN Library for Visual Studio 2005 (Microsoft Document Explorer – входит в состав дистрибутива VS. Нужно обязательно развернуть при установке VS!). Или online в Интернет.
5. С.О.Бочков, Д.М.Субботин Язык программирования Си для персонального компьютера, М.: "Радио и связь", 1990.- 384 с.
6. Фридланд А.Я. Информатика и компьютерные технологии: Основные термины: Толк.слов.: Более 1000 базовых понятий и терминов. – 3-е изд., испр. и доп./ А.Я Фридланд, Л.С. Ханамирова, И.А. Фридланд – М.:ООО «Издательство Астрель»: ООО «Издательство АСТ», 2003. - 272 с.

Дополнительная литература

7. Общее методическое пособие по курсу для выполнения ЛР и КЛР/ДЭ (см. на сайте 1-й курс www.sergebolshakov.ru) – см. кнопку в конце каждого раздела сайта!!!
8. Керниган Б., Ритчи Д. К Язык программирования С.\Пер. с англ., 3-е изд., испр. - СПб.: "Невский Диалект", 2001. - 352 с.: ил.
9. Другие методические материалы по дисциплине с сайта www.sergebolshakov.ru.
10. Конспекты лекций по дисциплине “Основы программирования”.
11. Подбельский В.В. Язык Си++: Учебное пособие. – М.: Финансы и статистика, 2003. (Доступно на сайте www.4read.org/2007/04/18/podbelskijj_v_jazyk_s.html)
12. 5. Подбельский В.В. Стандартный Си++: Учебное пособие. – М.: Финансы и статистика, 2008.
13. Г. Шилдт “С++ Базовый курс”: Пер. с англ.- М., Издательский дом “Вильямс”, 2011 г. – 672с
14. Г. Шилдт “С++ Руководство для начинающих” : Пер. с англ. - М., Издательский дом “Вильямс”, 2005 г. – 672с
15. Г. Шилдт “Полный справочник по С++”: Пер. с англ.- М., Издательский дом “Вильямс”, 2006 г. – 800с
16. Бьерн Страуструп "Язык программирования С++"- М., Бином, 2010 г.

Ссылки на размещение некоторых книг в Интернет вы найдете на сайте www.sergebolshakov.ru.

12. 12. Приложение:

12.1. Математическая библиотека

Из математической библиотеки можно использовать следующие основные функции:

- $\text{acos}(X)$ - возвращает значение арккосинуса аргумента
- $\text{asin}(X)$ - возвращает значение арксинуса аргумента
- $\text{atan}(X)$ - возвращает значение арктангенса аргумента
- $\text{cos}(X)$ - возвращает значение косинуса аргумента
- $\text{exp}(X)$ - возвращает значение экспоненты
- $\text{abs}(X)$ - возвращает абсолютное значение аргумента
- $\text{log}(X)$ - возвращает значение натурального логарифма аргумента

ОП ГУИМЦ 2024 ЛР№1

- $\log_{10}(X)$ - возвращает значение логарифма аргумента по основанию 10
- $\text{pow}(X, Y)$ - возвращает значение X , возведенное в степень Y
- $\sin(X)$ - возвращает значение синуса аргумента

В последних версиях VS допускается использовать аргументы и возвраты типа float и double.

Константы объявленные в файле <math.h>.

```
#define M_E          2.71828182845904523536      e
#define M_LOG2E      1.44269504088896340736
#define M_LOG10E     0.434294481903251827651
#define M_LN2        0.693147180559945309417
#define M_LN10       2.30258509299404568402
#define M_PI         3.14159265358979323846      π
#define M_PI_2       1.57079632679489661923      π/2
#define M_PI_4       0.785398163397448309616
#define M_1_PI       0.318309886183790671538
#define M_2_PI       0.636619772367581343076
#define M_2_SQRTPI   1.12837916709551257390
#define M_SQRT2      1.41421356237309504880
#define M_SQRT1_2    0.707106781186547524401
```