Филиал профессионального образовательного учреждения «Международный Открытый Колледж Современного Управления имени М.М. Абрекова» в Зеленчукском районе

ОДОБРЕНО
на заседании Педагогического совета ФПОУ «МОКСУ им. М.М. Абрекова»
Протокол № 1
от «24» июня 2023г.

«УТВЕРЖДАЮ»
Руководитель ФПОУ «МОКСУ им. М.М. Абрекова»
М.К. Хубиева
Приказ № 245
от «24» июня 2023г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
учебной дисциплины
ОПЦ.04 «Основы аллгоритмизации и программирования»

Специальность: *09.02.07. Информационные системы и программирование (базовая подготовка)*
Квалификация выпускника: *Программист*
Форма обучения: *очная*

Обсуждено
на заседании цикловой комиссии
«22» июня 2023 г.
Протокол № 1
Составитель программы:
Блимготова М.Т.

ст. Зеленчукская, 2023г.

Назначение фонда оценочных средств

Фонд оценочных средств (ФОС) составляется в соответствии с требованиями ФГОС СПО для проведения промежуточной аттестации обучающихся по дисциплине «Основы алгоритмизации и программирования» на соответствие их учебных достижений поэтапным требованиям соответствующей основной профессиональной образовательной программы (ОПОП). ФОС является составной частью рабочей программы дисциплины.

Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине «Основы алгоритмизации и программирования» включает в себя: перечень компетенций с указанием этапов их формирования в процессе освоения ОПОП; описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания; типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения ОПОП; методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций.

Фонд оценочных средств сформирован на основе ключевых принципов оценивания:

- валидности: объекты оценки должны соответствовать поставленным целям обучения;
- надежности: использование единообразных стандартов и критериев для оценивания достижений;
- объективности: разные студенты должны иметь равные возможности добиться успеха.

Основными параметрами и свойствами ФОС являются:

- предметная направленность (соответствие предмету изучения конкретной учебной дисциплины);
- содержание (состав и взаимосвязь структурных единиц, образующих содержание теоретической и практической составляющих учебной дисциплины);
- объем (количественный состав оценочных средств, входящих в ФОС);
- качество оценочных средств и ФОС в целом, обеспечивающее получение объективных и достоверных результатов при проведении контроля с различными целями.

I. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ С УКАЗАНИЕМ ЭТАПОВ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

1.1 Перечень формируемых компетенций

ПК	**ПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ**
ПК-1.1	Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
ПК-1.2	Разрабатывать программные модули в соответствии с техническим заданием.
ПК 1.3.	Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4.	Выполнять тестирование программных модулей.
ПК 1.5.	Осуществлять рефакторинг и оптимизацию программного кода.
ПК 1.6.	Разрабатывать модули программного обеспечения для мобильных платформ.

КОМПОНЕНТНЫЙ СОСТАВ КОМПЕТЕНЦИЙ

Код и формулировка компетенции	Компонентный состав компетенции		
	знать:	уметь:	иметь практический опыт:
ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.	**З1** – формы и способы взаимодействия со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности.	**У1** - Взаимодействовать со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности.	**ПО1 –** взаимодействия со специалистами смежного профиля при разработке методов, средств и технологий применения объектов профессиональной деятельности.
ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием. .	**З1 –** принципы управления ресурсами в информационной системе; **З2 –** основные задачи администрирования ИС; **З3** - основные понятия, функции, состав и принципы работы операционных систем;	**У1** - выполнять конфигурирование аппаратных устройств; **У2** - управлять учетными записями, настраивать параметры рабочей среды пользователей; **У3** - управлять работой информационной системы; управлять учетными записями, настраивать параметры рабочей среды пользователей.	**ПО1** - отладки программных модулей; **ПО2** - использования специализированных программных средств в процессе отладки программных модулей; **ПО3** - подготовки оборудования к работе.
ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.	**З1 –** методики и принципы формирования новых подходов для решения задач при работе в коллективе.	**У1** - формулировать основные положения и задачи для проведения исследований и обсуждения результатов коллективной деятельности	**ПО1** - необходимый для активного общения с коллегами из коллектива; коллективного обсуждения результатов работы, формирования новых коллективных подходов в решении задач.
ПК 1.4. Выполнять тестирование программных модулей.	**З1** – основные концепции, средства и особенности типичных представителей современных языков программирования; **З2** – основные понятия и принципы объектного подхода в информа-	**У1** – использовать знания о современных языках программирования в профессиональной деятельности; **У2** – использовать объектно-ориентированный анализ при проектировании информационных систем.	**ПО1** – программирования на современных языках программирования; **ПО2** – использования современных языков программирования для решения задач профессиональной деятельно-

	ционном моделировании.		сти.
ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.	**З1**- виды источников информации для профессиональной деятельности;	**У1**- организовывать эффективный поиск необходимой информации;	**ПО1** - использования различных источников, включая электронные;
ПК 1.6. Разрабатывать модули программного обеспечения для мобильных платформ.	**З1**- оборудование и инвентарь, используемое в области организации процесса производства;	**У1**- пользоваться необходимым оборудованием и инвентарем;	**ПО1** - безопасного использования новейшего оборудования;

I.ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ

1.1.Паспорт фонда оценочных средств

№ п/п	Наименование темы	Код контролируемой компетенции (или ее части)	Наименование контрольно-оценочного средства Текущий контроль/ баллы
1	2	3	4
1.	Тема 1. Основы аглоритмизации	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5
2.	Тема 2. Введение в программирование на языке C#	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5
3.	Тема 3. Операторы языка прогграммирования	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5
4.	Структура и классы. Файлы	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5
5.	Функции и процедуры	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5
6.	Модульное программирование	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5
7.	Объективно-ориентированное программирование (ООП)	ОК1. ,ОК 2.,ОК 3. ПК 4.1, ПК 4.2	Тест/5 Дискуссия/5 Доклад/5

III ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НАОБХОДИМЫЕ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОСОВЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Тема 1: «Решения задач на ЭВМ. Основные понятия алгоритмов»

Вопросы для устного обсуждения:

1. *Этапы решения задач. Понятие алгоритма*.
   Основные этапы решения задач на ЭВМ: постановка задачи, мат. или инф. Моделирование, разработка или выбор алгоритма, программирование.
2. *Свойства и формы записи алгоритмов*.
   Основные свойства алгоритмов. Словесный, формально-словесный, графический и программный способ записи алгоритмов.
3. *Линейные алгоритмы. Ветвления в алгоритмах.*
   Линейные и разветвляющиеся алгоритмы. Полная и неполная формы ветвлений.
4. *Циклические алгоритмы.*
   Счетный и итерационные циклы. Вложенные циклы.
5. *Понятие алгоритмических машин Поста и Тьюринга*.
   Основные элементы машины Поста. Система команд машины Поста. Машина Тьюринга.

Индивидуальные задания по теме:

Задание 1.

Цепочка из трех бусин, помеченных латинскими буквами, формируется по следующим правилам. В конце цепочки стоит одна из бусин A, B, C. На первом месте одна из бусин B, D, C, которой нет на третьем месте. В середине — одна из бусин A, C, E, B, не стоящая на первом месте. Какая из перечисленных цепочек создана по этим правилам?

1)CBB 2)EAC 3)BCD 4)BCB

Задание 2.

Определить значения переменных a и в после исполнения данного алгоритма, построив трассировочную таблицу.

a=11

a=112-a*9

в=51-33+a*2

a=a-2

a=a-b/a+39

где / - операция целочисленного деления.

Задание 3.

Сколько раз выполнится тело цикла, чему равны значения переменных a и в после выполнения фрагмента алгоритма?

Задание 4.

У исполнителя Утроитель две команды, которым присвоены номера:

1. Вычти 1

2. Умножь на 3

Первая из них уменьшает число на экране на 1, вторая - утраивает его. Запишите порядок команд в алгоритме получения из числа 5 числа 26, содержащем не более 5 команд, указывая лишь номера команд.

Задание 5.

Алгоритм из одной цепочки символов получает новую цепочку следующим образом. Сначала записывается буква, следующая в алфавите после первой буквы исходной цепочки, затем две последние буквы исходной цепочки в обратном порядке и, наконец, первая буква исходной цепочки. Получившаяся цепочка является результатом работы алгоритма. Например, если исходная цепочка символов была СОН, то результатом алгоритма будет цепочка ТНОС.

Дана цепочка символов КОШ. Какая цепочка символов получится, если к данной цепочке применить алгоритм трижды (т.е. алгоритм применяется к данной цепочке, затем к результату его работы, а затем ко второму результату работы алгоритма)?

Русский алфавит:

АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Тестирование:

1. Алгоритм, в котором команды выполняются в порядке их записи, то есть последовательно друг за другом, называется ...

a) линейным
b) прямым
c) простым

2. Выберете типы исполнителей

a) общие
b) формальный
c) неформальный

3. Алгоритм, в котором команды выполняются в зависимости от выполнения или невыполнения некоторого условия:

a) циклический
b) линейный
c) с ветвлением
d) линейный и циклический

4. Алгоритм, содержащий циклы, называется ...
a) линейным
b) алгоритм с повторение
c) ветвлением

5. Алгоритм, записанный на языке, понятном исполнителю, называется ...
a) записью
b) программой

6. Разработчиком алгоритма является ...
a) компьютер
b) человек
c) исполнитель

7. Какие формы записи алгоритмов вы знаете
a) графический
b) псевдокод
c) словесный
d) рисунчатый
e) программный

8. Соотнесите название фигуры с обозначением действия алгоритма
a) прямоугольник
b) овал
c) ромб
d) параллелограмм

1. выполняемое действие
2. условие для принятия решения
3. начало или конец
4. ввод или вывод

5. Расположите последовательно действия в алгоритме чистки обуви
a) Взять щетку и крем для обуви.
b) Вымыть и высушить обувь.
c) Взять обувь.
d) Выйти на лестницу.
e) Натереть обувь щеткой.
f) Принести обувь, щетку и крем в квартиру.
g) Намазать кремом обувь.

6. Компьютер работает в режиме:
a) формального управления
b) программного управления
c) неформального управления
d) непосредственного управления

7. Описание конечной последовательности шагов в решении задачи, приводящих от исходных данных к требуемому результату - это

a) алгоритм
b) программа
c) исполнитель

8. Некоторый объект (человек, животное, техническое устройство), способный выполнять определенный набор команд - это
a) компьютер
b) собака
c) исполнитель

9. Что можно считать алгоритмом?
a) Кулинарный рецепт
b) Перечень обязанностей дежурного по классу
c) Список класса
d) Правила техники безопасности

10. Фигура, служащая для обозначения последовательности действий в блок-схеме:
a) черта
b) ромб
c) звездочка
d) стрелка

11. Формальный исполнитель:
a) ученик
b) магнитофон
c) преподаватель
d) робот
e) животное
f) микроволновая печь

Тема 2: «Языки программирования и их назначение. Жизненный цикл программ. Основные этапы решения задач на компьютере»

Вопросы для устного обсуждения:

1. *Классификация ЯП.*
   Языки программирования низкого, высокого и сверхвысокого уровней.
2. *Основные понятия алгоритмических языков программирования.*
   Понятия алфавит, синтаксис, семантика. Трансляция программ. Компиляторы и интерпретаторы.
3. *Алфавит языка С++.*
   Символы, используемые для составления идентификаторов, разделители, специальные символы, зарезервированные слова.
4. *Общая структура программ в DevCpp*
   Декларационная часть, раздел функций, раздел основного блока программы. Комментарии в программах.

Индивидуальные задания по теме:

Задание 1.
В формировании цепочки из четырех бусин используются некоторые правила. В конце цепочки стоит одна из бусин P, N, T, O. На первом — одна из бусин P, R, T, O*;* которой нет на третьем месте. На третьем месте — одна из бусин O, P, T, не стоящая в цепочке последней. Какая из перечисленных цепочек могла быть создана с учетом этих правил?
1)PORT 2)TTTO 3)TTOO 4)OOPO

Задание 2.
Определить значения переменных a и в после исполнения данного алгоритма, построив трассировочную таблицу.
a=24
b=18
a=a+b/3+16
b=a+2*b
a=b/a+b/2
где / - операция целочисленного деления.

Задание 3.
Сколько раз выполнится тело цикла, чему равны значения переменных a и в после выполнения фрагмента алгоритма?

Задание 4.
У исполнителя Утроитель две команды, которым присвоены номера:
1. Вычти 2
2. Умножь на 3
Первая из НИХ уменьшает число на экране на 2, вторая - утраивает его. Запишите порядок команд в программе получения из 11 числа 13, содержащей не более 5 команд, указывая лишь номера команд.

Задание 5.
Алгоритм из одной цепочки символов получает новую цепочку следующим образом. Сначала записываются первые две буквы исходной цепочки в обратном порядке, затем буква, следующая в алфавите за последней буквой исходной цепочки, и, наконец, исходная цепочки символов, записанная в обратном порядке. Получившаяся цепочка является результатом работы алгоритма. Например, если исходная цепочка символов была СОН, то результатом алгоритма будет цепочка ОСОНОС.

Дана цепочка символов НИЛ. Какая цепочка символов получится, если к данной цепочке применить алгоритм дважды (т.е. алгоритм применяется к данной цепочке, затем к результату его работы)?
Русский алфавит:
АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Тема 3: «Основные понятия языка программирования С++. Простые типы данных языка С++. Совместимость типов».
Вопросы для устного обсуждения:

1. Классификация типов данных языка С++.
   простые и структурированные типы данных.
2. Порядковые типы данных.
   Целые типы, Символьный тип, Логические типы, Тип – диапазон. Функции применяемые к переменным целого типа.
3. Вещественные типы.
   Внутреннее представление вещественных чисел. Вещественные типы языка С++.
4. Процедуры ввода/вывода данных языка С++. Форматный вывод данных.
   Вывод данных вещественного типа.

Индивидуальные задания по теме:

Задание 1.
Для составления цепочек используются бусины, помеченные буквами A, B, C, D, E. На первом месте в цепочке стоит одна из бусин A, C, E. На втором — любая гласная, если первая буква согласная, и любая согласная, если первая гласная. На третьем месте — одна из бусин C, D, E, не стоящая в цепочке на первом месте. Какая из перечисленных цепочек создана по этим правилам?
1)CBE 2)ADD 3)ECE 4)EAD

Задание 2.
Определить значения переменных a и в после исполнения данного алгоритма, построив трассировочную таблицу.
a=15
b=2*a+24
a= b/2+a
в=2*a+2*b+9
a=b/a+2*a
где / - операция целочисленного деления.

Задание 3.
Сколько раз выполнится тело цикла, чему равны значения переменных I и A после выполнения фрагмента алгоритма?

Задание 4.
У исполнителя Калькулятор две команды, которым присвоены номера:
1. Прибавь 2
2. Умножь на 3
Выполняя первую из них, Калькулятор прибавляет к числу на экране 2, а выполняя вторую, утраивает его. Запишите порядок команд в программе получения из 0 числа 28, содержащей не более 6 команд, указывая лишь номера команд.

Задание 5.

Алгоритм из одной цепочки символов получает новую цепочку следующим образом. Сначала записывается буква, предшествующая в алфавите первой букве исходной цепочки, затем исходная цепочка символов, записанная в обратном порядке. Получившаяся цепочка является результатом работы алгоритма. Например, если исходная цепочка символов была СОН, то результатом алгоритма будет цепочка РНОС.
Дана цепочка символов РОК. Какая цепочка символов получится, если к данной цепочке применить алгоритм трижды (т.е. алгоритм применяется к данной цепочке, затем к результату его работы, а затем ко второму результату работы)?
Русский алфавит:
АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Практическая работа:

ТЕМА: «Составление и отладка программ с операторами ввода-вывода»

Цель работы:

- Закрепление навыков построения программ простейшей структуры;
- Формирование навыков в организации ввода/вывода значений стандартных типов данных;
- Получение практических навыков работы в диалоговом режиме.

Содержание работы:

Организовать ввод данных заданных типов. Вывести данные в указанных форматах, снабдив вывод соответствующими заголовками. Организовать распечатку вывода данных.

Даны: 3 целых числа, ширина поля – 6

6 вещественных чисел, ширина поля – 5, количество знаков после точки – 2

1 булевская переменная, количество символов – 4, значение переменной – FALSE

ЗАДАНИЯ:

№1. Составить программу, которая позволяет получить следующий вид экрана: Добрый день!

Как дела?

№2. Написать программу, которая выводит на экран четверостишие:

Я вас любил безмолвно, безнадежно,
То робостью, то ревностью томим;
Я вас любил так искренно, так нежно,
Как дай вам бог любимой быть другим.
А. С. Пушкин

№3. Написать программу которая вычисляет произведение 3 на 3 и выводит на экран в следующем виде:

Результат равен 9.

№4. Составить блок-схему и программу для вычисления площади прямоугольника со сторонами a=5 и в=14.

№5. Составить программу:

a) Для перевода верст в метры(1 верста=1066,8м.)
b) Для перевода минуты в секунды
c) Для перевода фунтов в граммы(1 фунт= 409,5гр.)

№6. Составить программу на нахождение произведения цифр четырехзначного числа.

№7. Необходимо определить сколько будет затрачено денег чтобы приобрести обои для проклейки стены длина которой А и высота В. Известно что рулон имеет длину 12м и ширину 1м. Цена за рулон N руб.

№8. Написать программу вычисления стоимости покупки, состоящей из нескольких ручек и книг.

№9. Написать программу нахождения суммы цифр трехзначного числа.

№10. Напишите программу которая меняет два значения местами, и выводит их на экран.

Тема 4: «Программирование разветвленных алгоритмов. Условная операция. Оператор выбора»

Вопросы для устного обсуждения:

1. *Классификация операторов.*
   Понятие оператора. Простые и структурные операторы.
2. *Операторы ветвления.*
   Общий вид и способ выполнения операторов if then и оператора switch.

Индивидуальные задания по теме:

Задание 1.
Цепочка из трех бусин формируется по следующим правилам. На первом месте в цепочке стоит одна из бусин А, Б, В. На втором — одна из бусин Б, В, Г. На третьем месте — одна из бусин А, В, Г, не стоящая в цепочке на первом или втором месте. Какая из следующих цепочек создана по этим правилам?
1)АГБ 2)ВАГ 3)БГГ 4)ББГ

Задание 2.
Определить значения переменных а и в после исполнения данного алгоритма, построив трассировочную таблицу.
a=48
b=a/4-5
a= a-b
b=2*a+37
a=b/a+2*a
где / - операция целочисленного деления.

Задание 3.

Сколько раз выполнится тело цикла, чему равны значения переменных I, A и R после выполнения фрагмента алгоритма?

Задание 4.
Исполнитель Калькулятор имеет только две команды, которым присвоены номера:
1. Вычти 3
2. Умножь на 2
Выполняя команду номер 1, Калькулятор вычитает из числа на экране 3, а выполняя команду номер 2, умножает число на экране на 2. Напишите программу , содержащую не более 5 команд, которая из числа 5 получает число 25. Укажите лишь номера команд.

Задание 5.
Алгоритм из одной цепочки символов получает новую цепочку следующим образом. Сначала записывается буква, предшествующая в алфавите последней букве исходной цепочки, затем первые две буквы исходной цепочки, и наконец, буква, следующая в алфавите за первой буквой исходной цепочки символов. Получившаяся цепочка является результатом работы алгоритма. Например, если исходная цепочка символов была СОН, то результатом алгоритма будет цепочка МСОТ.
Дана цепочка символов АРТ. Какая цепочка символов получится, если к данной цепочке применить алгоритм дважды (т.е. алгоритм применяется к данной цепочке, затем к результату его работы)?
Русский алфавит:
АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Практическая работа:

ТЕМА: «Составление и отладка программ с оператором if then»

Цель работы:

- Закрепление навыков построения блок-схем с ветвлениями;
- Закрепление навыков перевода алгоритма на язык программирования;
- Формирование навыков написания и отладки программ с ветвлениями.

Содержание работы:

Задание: Вычислить значение функции Y для любого, заданного пользователем, значения аргумента X

1. y= $\begin{cases} \frac{1}{a+b}, при\ a \le b \\ a+\sqrt{b}, при\ a > b \end{cases}$

a= $\begin{cases} \cos(x), при\ x \ge 0 \\ \sin(x), при\ x < 0 \end{cases}$

b= $\sqrt{\frac{1}{x+1}}$

2. y= $\sqrt{a+b}$; a = $\frac{1}{x+5}-\frac{1}{x-9}$; b = $\begin{cases}\cos(x), при\ x \geq 0 \\ \sin(x), при\ x < 0\end{cases}$

ТЕСТИРОВАНИЕ:

1. Языком высокого уровня является
   a) Макроассемблер
   b) Фортран
   c) Ассемблер

2. Как называется набор однотипных данных, имеющий общее для всех своих элементов имя
   a) Массив
   b) Множество
   c) запись

3. Раздел переменных определяется служебным словом
   a) TYPE
   b) VAR
   c) LABEL

4. Для возведения в квадрат применяется функция
   a) SQR(X)
   b) LOG(X)
   c) SQRT(X)

5. Какие из приведенных ниже типов данных относятся к вещественному типу данных
   a) real, single, extended
   b) word, double
   c) byte, real

6. Что выполняет операция ++ в языке С ++
   a) увеличивает значение операнда на единицу
   b) уменьшает значение операнда на единицу
   c) увеличивает значение операнда на два

7. Выберите, какой метод применяется для поиска в упорядоченных массивах
   a) прямой обмен

b) бинарный поиск
c) прямой выбор

8. Выясните, в основе какого метода сортировки лежит обмен соседних элементов массива
a) прямой выбор
b) прямой обмен
c) оба ответа верны

9. В Паскале различие в высоте букв (прописные или строчные)
a) имеет значение для написания имен переменных
b) имеет значение для написания служебных слов
c) имеет значение при написании текстовых констант

10. В сложных выражениях последовательность выполнения операций определяется
a) только скобками
b) скобками, приоритетом операций, а при одинаковом приоритете ассоциативностью операций
c) только приоритетом операций

11. В сложных выражениях последовательность выполнения операций определяется
a) только скобками
b) только приоритетом операций
c) скобками, приоритетом операций, а при одинаковом приоритете ассоциативностью операций

12. Определите, как называется процесс перестановки элементов массива с целью упорядочивания их в соответствии с каким-либо критерием
a) сортировка
b) поиск
c) перебор

13. Выясните, в основе какого метода сортировки лежит обмен соседних элементов массива
a) прямой выбор
b) прямой обмен
c) оба ответа верны

14. Разработке алгоритма предшествует
a) выбор метода решения
b) выбор метода решения
c) разработка математической модели

15. Какой вид подпрограмм есть в Паскале
a) модуль
b) процедура
c) размер

16. В чем разница между фактическими и формальными параметрами функций
a) и формальные, и фактические параметры используются вне тела функции

b) нет разницы, это одни и те же параметры
c) формальные параметры определены в теле функции, а фактические — значение, с которыми функция вызывается

17. Вещественный тип данных объявляется служебным словом
    a) INTEGER
    b) DOUBLE
    c) LONGINT

18. Записью действительного числа с плавающей точкой является
    a) 48.0001
    b) –1.0533333
    c) 1.0E01

19. Для вычисления экспоненты применяется процедура
    a) EXP(X)
    b) TRUNC(X)
    c) SQR(X)

20. Какие из приведенных типов данных относятся к целочисленному типу данных
    a) integer, word, longint
    b) integer, real
    c) comp, double

Тема 5: «Программирование циклических алгоритмов».

Вопросы для устного обсуждения:

1. *Оператор цикла с параметром* ***(for…do)***;
   назначение, общий вид и способ выполнения оператора **for…do.**
2. *Оператор цикла с предусловием* ***(while…do)***;
   Назначение, общий вид, способ выполнения и блок-схема оператора while…do. Примеры использования.
3. *Оператор цикла с постусловием* ***(do…while***).
   Назначение, общий вид, способ выполнения и блок-схема оператора опрератора цикла с постусловием do…while. Примеры использования.
4. *Вложенные циклы.* Понятие вложенных циклов. Глубина вложенности.
5. *Процедуры управления циклами*. Использование оператора продолжения цикла CONTINUE и оператора BREAK.
6. *Примеры составления программ с использованием итераций, ветвлений, повторений.*

   Нахождение суммы сходящегося ряда, табулирование функции на заданном отрезке и т.д.

Индивидуальные задания по теме:

Выбрать функцию по варианту из таблицы.
Составить программу расчета таблицы значений функции f(x)
на интервале a<=x<=b в n равностоящих точках. Границы интервала
a, b и количество точек n ввести с клавиатуры. Результаты вывести
на печать.

Вариант	Функция
1	$f(x) = 5 \cdot (1 - e^{-0.5x}) \cdot \cos(2\pi x)$ Найти сумму всех положительных значений функции $f(x)$ в расчетных точках
2	$f(x) = 4 \cdot e^{-0.5x} \cdot \cos(\pi x)$ Найти сумму всех отрицательных значений функции $f(x)$ в заданном интервале
3	$f(x) = 5 \cdot e^{-0.5x} \cdot \sin(\pi x)$ Найти произведение вычисленных значений функции $f(x)$, целая часть которых кратна 2
4	$f(x) = (1 - e^{-x}) \cdot \sin(4\pi x)$ Найти среднее арифметическое положительных значений функции $f(x)$
5	$f(x) = 1 + \sin(2\pi x)$ Найти количество положительных значений функции $f(x)$ на заданном интервале
6	$f(x) = 4 \cdot e^{-0.5x} \cdot \cos(\pi x)$ Найти произведение всех отрицательных значений функции $f(x)$ на заданном интервале
7	$f(x) = 5 \cdot (1 - e^{-0.5x}) \cdot \cos(2\pi x)$ Найти произведение всех положительных значений функции $f(x)$ на заданном интервале
8	$f(x) = (1 - e^{-x}) \cdot \sin(4\pi x)$ Найти сумму вычисленных значений функции $f(x)$, целая часть которых кратна 2
9	$f(x) = 1 + \frac{\sin 2\pi x}{1 + x}$ Найти среднее арифметическое отрицательных значений функции $f(x)$
10	$f(x) = \frac{\cos(4\pi x)}{1 + x^2}$ Найти сумму вычисленных значений функции $f(x)$, дробная часть которых > 0,5

Темы рефератов:

1. История и классификация языков программирования.
2. Системы программирования.
3. Процедурные и непроцедурные языки программирования.
4. Объектно-ориентированное программирование.
5. Язык программирования C++. История создания. Использование в современности.
6. Биография Бьярне Страуструпа.

Тема 6: «Структурированные типы данных языка С++. Массивы, двумерные массивы. Основные операции над массивами»

Вопросы для устного обсуждения:

1. *Описание массивов. Ввод и вывод массивов.* Обращение к элементу массива. Определение массива. Описание одномерных массивов, базовый тип элементов массива. Поэлементный ввод и вывод массивов.
2. *Методы сортировки одномерных массивов.* Алгоритмы метода прямого перебора элементов массива и метода прямого обмена («пузырьковый» метод»).
3. *Двумерные массивы. Алгоритмы обработки матриц.* Описание двумерных массивов. Вывод матрицы в виде таблицы. Типовые алгоритмы обработки матриц.

Тестовые задания:

1. Отметьте все правильные утверждения о массивах в языке С++

a) элементы массива могут быть разных типов
b) все элементы массива должны быть одного типа
c) элементы в памяти расположены рядом
d) элементы могут быть расположены в памяти по одному
e) элементы всегда нумеруются с нуля

2. Какой индекс имеет последний элемент массива A?

```
int A[6];
```

Ответ:

3. Требуется заполнить массив именно так:

X = [1 3 5 7 9 11]

Какой оператор надо поместить в тело цикла вместо многоточия?

```
for ( k=0; k<6; k++ ) {
  ...
}
```

a) X[k] = k
b) X[k] = 2*k
c) X[k] = 2*k - 1
d) X[k] = 2*k + 1
e) X[k] = 2*(k + 1)

4. Требуется заполнить массив именно так:

X = [12 9 6 3 0 -3]

Какой оператор надо поместить в тело цикла вместо многоточия?

```
for ( k=0; k<6; k++ ) {
 ...
}
```

a) X[k] = k
b) X[k] = 12 - 2*k
c) X[k] = 3*k - 12

d) X[k] = 3*(k + 1) + 9
e) X[k] = 12 - 3*k

5. Требуется заполнить массив именно так:

X = [0 3 4 7 8 11]

Какой оператор надо поместить в тело цикла вместо многоточия?

```
for ( k=0; k<6; k++ ) {
  ...
}
```

a) X[k] = 3*k - k % 2
b) X[k] = 2*k + k % 2
c) X[k] = 2*k - k % 2
d) X[k] = 2*k + k / 2
e) X[k] = 2*(k - 1)

6. Требуется заполнить массив именно так:

X = [1 2 4 8 16 32]

Какой оператор надо поместить в тело цикла вместо многоточия?

```
X[0] = 1;
for ( k=1; k<6; k++ ) {
  ...
}
```

a) X[k] = k
b) X[k] = 2*k
c) X[k] = X[k-1] + 1
d) X[k] = 2*X[k-1]
e) X[k] = 2*(X[k-1] - 1)

7. Что надо написать вместо многоточия, чтобы вывести элементы массива X[N] в обратном порядке? В ответе не используйте пробелы.

```
for ( k=0; k<N; k++ )
  cout << ... << endl;
```

Ответ:

8. Какой оператор надо вставить вместо многоточия, чтобы вывести на экран все элементы массива A[N] с четными номерами? В ответе не используйте пробелы.

```
k = 0;
while ( k < N ) {
  cout << A[k] << " ";
  ...
}
```

Ответ:

9. Задан массив X[N]. Какой оператор надо поставить вместо многоточия, чтобы найти сумму всех элементов массива в переменной S? Вводите ответ без пробелов.

```
S = 0;
for ( k=0 k<N; k++ ) {
  ...
}
```

Ответ:

10. Задан массив X[N]. Какое условие надо поставить вместо многоточия, чтобы найти сумму положительных элементов массива в переменной S? Вводите ответ без пробелов.

```
S = 0;
for ( k=0; k<N; k++ )
   if ( . . . ) S = S + X[k];
```

Ответ:

11. Задан массив X[N]. Какое условие надо поставить вместо многоточия, чтобы найти количество четных элементов массива в переменной S?

```
S = 0;
for ( j=0; j<N; j++ )
if ( ... ) S++;
```

a) S / 2 == 1
b) S % 2 == 0
c) X[j] % 2 == 1
d) X[j] % 2 == 0
e) X[j] / 2 == 0

Тема 7: «Работа со строками и файлами.»

Контрольные вопросы:

1. Почему в С++ не выполняется операция прямого присваивания значения строке?
2. Почему символ и строка, состоящая из одного символа, занимают разный объем памяти?
3. Почему в функции scanf("%s",string); не указывается обращение к переменной по адресу?
4. Допустима ли операция сравнения над символами? Если да, то каким образом определены отношения "больше" и "меньше"?
5. Какая из функций, gets или puts, заносит в поток управляющий символ '\n' и с какой целью?
6. Можно ли выполнить присваивание символьной переменной числового значения? Почему?
7. В чем различия результатов вывода символьной переменной со спецификаторами %d и %c?

Лабораторная работа:

Цель работы: изучить функции работы со строками в С++.

Задание

С помощью текстового редактора создать файл, содержащий текст, длина которого не превышает 700 символов (длина строки не превышает 70 символов).
Имя файла должно иметь расширение .DAT.
Написать программу, которая:

- выводит текст на экран дисплея;
- по нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (вводится с клавы);

- выводит текст на экран дисплея еще раз, удаляя из него заданное слово и лишние пробелы.

Теоретические сведения:
Строка в C++ представляется как массив символов, завершающихся символом ‘\0’.
Основные функции для работы со строками библиотеки «STRING.H»:
strcpy(char * dest,const char *source) — копирует source в dest при этом не происходит проверки на выход за границу массива символов
strcat(char *dest,const char *source) — добавляет source в dest при этом не происходит проверки на выход за границу массива символов
strupr(char *str) — преобразует строку к заглавным буквам
int strlen(char *str) — вычисляет длину строки
char *strdup(const char *str) — создаёт дубликат строки в динамической памяти

Текст программы:

```
#include <fstream.h>
#include <stdio.h>
#include <process.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>

const unsigned int MAX=700;//700*71;//не более 70 символов
const MAX_WORD=20;
char * FileName="INPUT.DAT";

void ReadFile(ifstream & InputFile,char *buffer)
{
        InputFile.read((char*)buffer,MAX);
        buffer[InputFile.gcount()-1]='\0';


}

int SelectWord(char *buffer,char *word, int k,int mode)
{
        char *s,*ls;//Для нахождения подстроки
        char lc;

        ls=buffer;
        s=strstr(buffer,word);
        int n=0;
        int yes=0;
        while (s!=NULL)
        {
                lc=s[0];
                s[0]='\0';
                printf(ls);
                if (n==k) {textcolor(GREEN);yes=1;}
if (mode==0)  cprintf("%s",word);
```

```
                n++;
                textcolor(LIGHTGRAY);
                ls=s+strlen(word);
                if (mode==1)
                {
                        if ((ls[0]!='\0')||(ls[0]!='\n')) ls++;
                }
                s[0]=lc;
                s=strstr(ls,word);
        }
        printf("%s",ls);
        return yes;
}


void main()
{
        printf("\n\nstartint lab6.\n\n");
        char *buffer=new char[MAX];
        char *word=new char[MAX_WORD];
        char *str1=new char[100];//Для внутренних нужд
        if ((buffer==NULL)||(!word)||(!str1))
        {
                printf("Не достаточно памяти\n");
                exit(1);
        }
        ifstream InputFile;
        InputFile.open(FileName,ios::in);
        if (InputFile.fail())
        {
                printf("Не удалось открыть файл \"%s\"\n",FileName);
                exit(1);
        }

        ReadFile(InputFile,buffer);
        strupr(buffer);
        clrscr();
  printf("Вывод файла:\n");
        printf("%s",buffer);
        ////////////////////////////
        //Читаем слово с клавиатуры
        printf("\nВведите слово для поиска: ");
        scanf("%20s",word);
        strupr(word);
        printf("\n");
        int k=0;
        clrscr();
        printf("Выделяю слово:\n");
```

```
        while (SelectWord(buffer,word,k,0))
        {
                getch();
                k++;
                clrscr();
                printf("Выделяю слово:\n");
        }
        InputFile.close();
        clrscr();
        printf("Тот же текст с удалённым словом:\n");
        SelectWord(buffer,word,k,1);
        getch();
        delete buffer;


}
```

Файл «INPUT.DAT»:

```
1Terminator is the main robot in the Universe.
2He a hero of a film called Terminator
3Kaban is a swine atrussian.
4Universe is endless.
5Film.
```

Вывод: Были изучены основные функции для работы со строками в C++.

Практическая работа:

1. Получить от пользователя строку текста и вставить вместо каждого пробела восклицательный знак. Повторять до тех пор, пока не встретится строка завершающаяся словом quit.
2. Открыть текстовый файл в необходимом режиме (на чтение, на чтение и запись, на добавление). (В файле должно быть не менее 30 строк). Подсчитать количество слов в каждой строке текста. Записать в новый файл все строки, закончив их полученным числом.

Например: Это пример строки текста в старом файле (7)

3. Описать структуру с именем AEROFLOT, содержащую следующие поля:

NUMR – номер рейса (целое число);
COST – стоимость билета (число с плавающей точкой);
NAZN – название пункта назначения рейса;
TIP – тип самолета.
Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив AIRPORT, состоящий из семи элементов типа AEROFLOT;
- вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры;

- если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Тема 8: «Препроцессорные средства. Память. Адреса. Указатели.»

Вопросы для устного обсуждения:

1. Назначение макросов?
2. Что такое макроподстановка?
3. Что такое именованная константа?
4. В чём состоит сущность условной компиляции?
5. Привести пример использования макрокоманд обработки переменного числа параметров.

Лабораторная работа:

1. Цель работы – изучить особенности работы с директивами препроцессора в C++.

2. Теоретические положения.

Препроцессор - это компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы (например, компилятора). О данных на выходе препроцессора говорят, что они находятся в препроцессированной форме, пригодной для обработки последующими программами (компилятор). Результат и вид обработки зависят от вида препроцессора; так, некоторые препроцессоры могут только выполнить простую текстовую подстановку, другие способны по возможностям сравниться с языками программирования. Наиболее частый случай использования препроцессора - обработка исходного кода перед передачей его на следующий шаг компиляции. Языки программирования C/C++ и система компьютерной вёрстки TeX используют препроцессоры, значительно расширяющие их возможности.

В некоторых языках программирования этап компиляции и трансляции получили название «препроцессинга».

Макрокоманда, макроопределение или макрос - программный алгоритм действий, записанный пользователем. Часто макросы применяют для выполнения рутинных действий.

Препроцессорная обработка (макрообработка) - это преобразование текста путем замены препроцессорных переменных их значениями и выполнения препроцессорных операторов (директив препроцессора).

В общем случае препроцессорные средства включают:

- определение препроцессорных переменных и присвоенных им
- значений;
- средства управления просмотром преобразуемого текста;
- правила подстановки значений макропеременных.

Директивы препроцессора представляют собой инструкции, записанные в тексте программы на C++, и выполняемые до трансляции программы. Все директивы препроцессора начинаются со знака #, которому могут предшествовать только пробелы и знаки табуляции. Если директива не помещается в одной строке, в конце строки ставится знак \ и директива продолжается на следующей строке. Количество строк продолжения не ограничивается. После директив препроцессора точка с запятой не ставятся.

2.1. Директива #include

Директива #include включает в текст программы содержимое указанного файла. Эта директива имеет две формы:

#include "имя файла" #include <имя файла>

2.2. Директива #define

Директива #define служит для замены часто использующихся констант, ключевых слов, операторов или выражений некоторыми идентификаторами.

Идентификаторы, заменяющие текстовые или числовые константы, называют именованными константами. Идентификаторы, заменяющие фрагменты программ, называют макроопределениями, причем макроопределения могут иметь аргументы.

Директива #define имеет две синтаксические формы:

#define идентификатор текст

#define идентификатор (список параметров) текст

Эта директива заменяет все последующие вхождения идентификатора на текст. Такой процесс называется макроподстановкой. Текст может представлять собой любой фрагмент программы на С, а также может и отсутствовать. В последнем случае все экземпляры идентификатора удаляются из программы.

Пример:

#define WIDTH 80

#define LENGTH (WIDTH+10)

Эти директивы изменят в тексте программы каждое слово WIDTH на число 80, а каждое слово LENGTH на выражение (80+10) вместе с окружающими его скобками.

Во второй синтаксической форме в директиве #define имеется список формальных параметров; при макровызове вслед за идентификатором записывается список фактических аргументов, количество которых должно совпадать с количеством формальных параметров.

Пример:

#define MAX(x,y) ((x)>(y))?(x):(y)

Эта директива заменит фрагмент t=MAX(i,s[i]);

на фрагмент

t=((i)>(s[i])?(i):(s[i]);

2.3.Директива #undef

Директива #undef используется для отмены действия директивы #define.

Синтаксис этой директивы следующий:

#undef идентификатор

Директива отменяет действие текущего определения #define для указанного идентификатора. Пример:

#undef WIDTH #undef MAX

Имеется ряд предопределенных макроимен, предусмотренных стандартами на языки С и С++, в том числе:

_ _LINE_ _

- номер строки в исходном файле,

_ _FILE_ _

- имя обрабатываемого файла,

_ _DATE_ _

- дата начала обработки препроцессором,

_ _TIME_ _

- время начала обработки,

_ _STDC_ _

- программа должна соответствовать стандарту ANSI.

_ _cplusplus

-компилировать программу в соответствии с синтаксисом Си++,

_ _PASCAL_ _

-последующие имена по умолчанию имеют тип

“имя языка Pascal”

Предопределенные имена нельзя объявлять в #define или отменять в

#undef.

2.4.Условная компиляция

Вызывает деление текста программы на части, которые компилируются (или не компилируются) в зависимости от некоторых внешних условий.

Начинается в одном из трех случаев:

#if константное_выражение /*Определено?*/

#ifdef идентификатор

/*Определен?*/

#ifndef идентификатор

/*Не определен?*/

Для #if вычисляется константное_выражение, и если результат вычислений - истина (ненулевой), то компилируются все строки программы вплоть до директивы

#else,

#elif

или

#endif.

Для #ifdef (

#ifndef) компилируется соответствующая часть кода, если определен (не определен) идентификатор.

Вконстантном_выражении может быть использована операция defined для проверки, был ли определен указанный идентификатор.

Обратное условие проверяет операция !defined.

Ранние версии препроцессора не имеют операции defined, а работают только с #ifdef и #ifndef. Эти директивы были сохранены для совместимости (кроме того, запись #ifdef короче, чем #if defined).

Может иметь раздел "иначе", определяемый в одной из двух форм:

#else

#elif константное_выражение

Для #else соответствующая последовательность кода компилируется в том случае, если не выполняется условие, указанное в #if.

#elif аналогично #else за исключением того, что должно еще выполняться условие, задаваемое константным_выражением.

Должна заканчиваться директивой #endif.

2.5.Директива #error

Директива #error останавливает работу компилятора и выдает сообщение об ошибке.

2.6. Директива #line

Директива #line константа “имя. файла” заставляет компилятор считать, что константа задает номер следующей строки исходного файла, и текущий входной файл именуется идентификатором. Если идентификатор отсутствует, то запомненное имя файла не изменяется.

2.7. Директива #pragma

#pragma – это директива препроцессора, которая реализует возможности компилятора. Эти особенности могут быть связанны с типом компилятора.

Разные типы компиляторов могут поддерживать разные директивы. Общий вид директивы:

#pragma команда компилятора

2.8. Макросы обработки переменного числа параметров

Макрокоманды находятся в заголовочном файле "stdarg.h". Тип va_list используется для переменной указателя аргумента: va_list (тип данных)

Следующая макрокоманда инициализирует переменную указателя аргумента ap, чтобы указать на первый из необязательных аргументов текущей функции; last_required должен быть последний требуемый аргумент функции:

void va_start (va_list ap, last_required) (макрос)

Va_arg макрокоманда возвращает значение следующего необязательного аргумента, и изменяет значение ap, чтобы указать на последующий аргумент.

Таким образом, последовательные использования va_arg возвращают последовательные необязательные аргументы.

type va_arg (va_list ap, type) (макрос)

Последняя макрокоманда заканчивает использование ap:

void va_end (va_list ap) (макрос)

Последовательность использования макрокоманд задана следующим образом: инициализация переменной указателя аргумента типа va_list, используя va_start. Указатель аргумента, после инициализации, указывает на первый необязательный аргумент; обращение к необязательным аргументам последовательными применением va_arg; окончание работы – вызовом va_end.

3. Задания

Разработать пошаговый алгоритм решения задачи. Выполнить программную реализацию задачи согласно варианту. В программе обязательно предусмотреть вывод информации об исполнителе работы (ФИО), номере варианта, выбранном уровне сложности и задании. Предусмотреть возможность повторного запуска программы (запрос о желании выйти из программы, или продолжить работу). Отслеживать все особые случаи входных данных.

Ключевые моменты программы обязательно должны содержать комментарии.

Подготовить отчет.

1. Используя один и тот же идентификатор объявить цифровой и символьный массивы. Для цифрового массива определить количество повторений каждого числа. Для символьного массива найти позицию и значение символа по заданному коду символа. Использовать условную компиляцию.

2. Проверить вводимые с клавиатуры числа на чётность и положительность. Оформить в виде макросов определение чётности,

положительности, вывод диагностических уведомлений. С помощью условной

компиляции разделить процессы определения чётности и положительности

чисел.

3. Используя макросы __FILE__, __LINE__, вывести информационные сообщения. Сформировать произвольное сообщение о программной ошибке.

4. Используя один и тот же идентификатор объявить цифровой и символьный массивы. Для цифрового массива найти значение максимального элемента. Для символьного массива найти количество слов (слова разделяются пробелами). Использовать условную компиляцию.

4. Требования к отчету

Отчет подается после полной сдачи и защиты лабораторной работы в электронном виде (документ Word).

Отчет должен быть оформлен согласно ДСТУ 3008-95.

В отчет должен содержать следующие пункты:

- титульный лист;
- содержание;
- цель работы;
- постановка задачи;
- аналитические выкладки, если необходимо;
- пошаговый алгоритм решения задачи;

- исходный код программы с комментариями;
- примеры работы программы;
- выводы.

Отчет оценивается максимально в 1 балл.

Индивидуальные задания по теме:

1. Используя макросы обработки переменного числа параметров создать программу сортировки чисел (использовать метод сортировки слиянием).

2. В отделе работают *n* сотрудников, которые получают заработную плату в гривнах. Требуется определить: насколько зарплата самого высокооплачиваемого из них отличается от самого низкооплачиваемого.

Решить задачу, используя макросы обработки переменного числа параметров.

3. Задано *n* натуральных чисел. Необходимо определить наибольшее натуральное число, отсутствующее в последовательности. Решить задачу,

используя макросы обработки переменного числа параметров.

4. Используя макросы обработки переменного числа параметров,

создать программу генерации сообщений с помощью передачи каждого слова как отдельного параметра.

5. Используя макросы обработки переменного числа параметров,

составить программу вычисления наибольшего общего делителя произвольного числа целых чисел.

6. Отрезки на плоскости задаются парами целочисленных координат концевых точек ($x1$,$y1$), ($x2$,$y2$). Определить, пересекаются ли 2 отрезка. Для расчетов использовать макросы вычислений с параметрами.

7. Задано *n* чисел, найти сумму всех чисел, на которые каждое *ni i 1, n*

делится без остатка, независимо друг от друга. Например: 4-7 6-12. Решить задачу, используя макросы обработки переменного числа параметров.

8. Дано *n* камней с массами *m1, m2 , … mn*. Требуется разложить камни на 2 кучки так, чтобы разница масс этих кучек была минимальной. Решить задачу, используя макросы обработки переменного числа параметров.

9. Используя макросы обработки переменного числа параметров создать программу сортировки чисел (использовать метод быстрой сортировки).

10. Будем называть числа дружными, если они состоят из одних и тех

же цифр. Например, числа 1132 и 32321 являются дружными, а 12 и 123 – нет.

Используя макросы обработки переменного числа параметров, определить являются ли *n* целых чисел дружными.

11. Задано *n* целых чисел и целое число *q*, используя макросы обработки переменного числа параметров, определить какие из чисел *ni i 1, n*

делятся без остатка на *q*.

12. Используя макросы обработки переменного числа параметров создать программу сортировки чисел (использовать метод сортировки пузырьком).

13. Определим простой цифровой корень (ПЦК) натурального числа *N*

следующим образом. Если *N* - простое число, то ПЦК(*N*) = *N*. Если число однозначное, но не простое (то есть 1, 4, 6, 8 или 9), то ПЦК(*N*) = 0. В

остальных случаях ПЦК(N) = ПЦК(S(*N*)), где S(*N*) - сумма цифр числа N. Найти ПЦК для *n* заданных чисел используя макросы обработки переменного числа параметров.

14. Задано *n* чисел *ni* 100 *i* 1, *n* . Удалить из каждого числа по две цифры, так чтобы получилось минимально возможное число. Решить задачу,

используя макросы обработки переменного числа параметров.

15. Задано *n* чисел, используя макросы обработки переменного числа параметров найти наименьшее число, для количеством делителей, или вывести -1, если такого числа нет. Например: 3-4 4-6.

16. Имеется ряд из *n* лампочек, которые пронумерованы от 1 до *n*.
Изначально ни одна из лампочек не горит. Далее происходит *k* последовательных линейных инверсий этого ряда ламп. Под линейной инверсией понимается инверсия каждой *p*-й лампочки в ряде. Например, если *p*=3, то произойдет инверсия 3-й, 6-й, 9-й и т.д. лампочек. Требуется определить: сколько горящих лампочек останется после реализации всех заданных линейных инверсий? Решить задачу, используя макросы обработки переменного числа параметров.

17. Задана строка, состоящая из нескольких слов разделенных пробелами и ширина листа, используя макросы обработки переменного числа параметров, записать слова строки в столбик по центру листа, центрирование выполнить при помощи вставки в конец и в начало слова пробелов. Пробелы для наглядности можно заменить другим символом, например «~».
Предусмотреть случай, когда выполнить подобное центрирование невозможно.

18. Имеются гири с массами: 1 г, 2 г, …, *n* г., используя макросы обработки переменного числа параметров распределить эти гири на максимально возможное количество пар так, чтобы суммарный вес гирь в каждой паре выражался простым числом. Вес каждой гири передавать как отдельный параметр.

19. Постулат Бертрана гласит, что для любого n > 1 найдется простое число p в интервале n < p < 2n. Такая гипотеза была выдвинута в 1845 году французским математиком Джозефем Бертраном и доказана в 1850 году Пафнутием Чебышевым.

Задача состоит в том, чтобы для ряда чисел, используя макросы обработки переменного числа параметров, решить несколько более общую задачу – а именно по числу n найти количество простых чисел p из интервала n
<p < 2n.

20. Задано *n* чисел. Используя макросы обработки переменного числа
параметров найти количество натуральных чисел, не превышающих *ni i* 1, *n* и
не делящихся ни на одно из чисел 2, 3, 5.

21. Рассмотрим дробь 1/*n, n*>1. Как известно, цифры, в её десятичной записи начиная с некоторого места повторяются. Минимальную по длине повторяющуюся (без промежутков) часть называют периодом. Минимальную по длине часть после запятой, которая не входит ни в один период, называют предпериодом. Например: *n*=28: 1/28=0,03(571428), предпериод '03', длина 2,
период '571428', длина 6.

Задано несколько дробей в виде «1/*n*», используя макросы обработки переменного числа параметров найти длину предпериода и длину периода у каждой дроби.

22. Важным понятием теории формальных грамматик и автоматов является формальный язык. Неформально его можно определить как некоторое множество слов, где под словом понимается некоторая строка из символов.

Задано несколько слов, необходимо определить какие из них принадлежат языку {0n1n2n, n ≥ 1}. В этот язык входят те и только те слова,
которые имеют такую структуру: в них нулей столько же, сколько единиц, а единиц - столько же, сколько и двоек. При этом любой ноль находится ближе к началу слова, чем любая единица, а любая единица находится ближе к началу слова, чем любая двойка.
Например, слово 001122 принадлежит этому языку, а
слово 0120 - не принадлежит.

Решить задачу, используя макросы обработки переменного числа параметров. Параметрами должны выступать отдельные слова.

23. Задано несколько слов состоящих из латинских букв. Необходимо выполнить свертку каждого слова по следующим правилам:

несколько подряд идущих одинаковых символов заменить числом и соответствующей буквой (aaa→3a);

несколько подряд идущих одинаковых последовательностей заменить числом и соответствующей последовательностью (abab→2ab);

Решить задачу, используя макросы обработки переменного числа параметров. Параметрами должны выступать отдельные слова.

24. Задано n чисел $ni\ 100\ i\ 1, n$ *. Удалить из каждого числа по две* цифры, так чтобы получилось максимально возможное число. Решить задачу, используя макросы обработки переменного числа параметров.

25. Задано *n* натуральных чисел. Необходимо определить наименьшее натуральное число, отсутствующее в последовательности. Решить задачу, используя макросы обработки переменного числа параметров.

26. Пусть слово - это последовательность от 1 до 10 символов, не включающая пробелов. Имеем *n* слов *A*1,...,*A*n. Переупорядочить слова так, чтобы получилась "цепочка", т.е. для каждого слова $Ai\ ,i\ 1,n$ его первая буква должна совпадать с последней буквой предыдущего слова, а последняя буква в $Ai\ ,i\ 1,n$ - с первой буквой последующего слова; соответственно последняя буква последнего слова должна совпадать с первой буквой первого слова. В цепочку входят все *n* слов без повторений. Если такое переупорядочивание невозможно – выдать соответствующее сообщение. Решить задачу, используя макросы обработки переменного числа параметров. Каждое слово выступает отдельным параметром.

27. Дано *n* косточек домино в виде $(ai / bi\ ),i\ 1,n$, по правилам игры выкладываются в прямую цепочку, начиная с косточки, выбранной произвольно, в оба конца до тех пор, пока это возможно. Нужно определить, возможно ли выложить все косточки домино в цепочку. Решить задачу, используя макросы обработки переменного числа параметров. Каждая косточка выступает отдельным параметром.

28. Задано *n* чисел $ni\ 10\ i\ 1,n$. Разделить числа на две группы, так чтобы отличия в произведении чисел в каждой группе были минимальны. Решить задачу, используя макросы обработки переменного числа параметров.

29. Задано *n* монет из различных стран с массой $mi\ i\ 1,n$. Монеты упорядочены по возрастанию массы монет. Найти минимальное число взвешиваний на чашечных весах, для определения места, куда поместить новую монету, с массой *M*, чтобы не нарушить последовательность. Решить задачу,используя макросы обработки переменного числа параметров.

30. Задано *n* слов разной длины. Необходимо упорядочить слова по алфавиту. Решить задачу, используя макросы обработки переменного числа параметров.

Тема 9: «Динамические структуры данных (списки) Формирование списков».

Вопросы для устного обсуждения:

1. В чем принципиальное отличие линейного однонаправленного (*двунаправленного*) и циклического однонаправленного (*двунаправленного*) списков?
2. Как избежать зацикливания при просмотре циклического списка?
3. Какое значение содержит указатель на *дек*?
4. Нужно ли в деке определять первый элемент? Ответ обоснуйте.
5. На основании чего в красно-черном дереве самая длинная ветвь от корня к листу не более чем вдвое длиннее любой другой ветви от корня к листу?
6. Куда может быть добавлен элемент в красно-черное дерево? Вид дерева при этом должен сохраниться.
7. Как можно охарактеризовать красно-черное дерево: полное, неполное, строгое, нестрогое?
8. Каким образом при удалении элемента из красно-черного дерева перекрашиваются узлы?

Лабораторная работа: Решение задач на динамические структуры данных.

Задачи:

Цель работы: изучить алгоритмы и приемы работы с *динамическими структурами данных*, научиться решать задачи с использованием *динамических структур данных* и алгоритмов работы с ними на языке С++.

При выполнении лабораторной работы для каждого задания требуется написать программу на языке С++, в которой выполнено формирование красно-черного дерева, *дека* или циклического списка в соответствии с постановкой задачи, ввод данных элементов динамических структур с учетом типа информационного поля, их обработка и *вывод* на экран в указанном формате. Для хранения данных динамических структур следует использовать ресурсы динамической памяти. Ввод данных осуществляется с клавиатуры с учетом требований к входным данным, содержащихся в постановке задачи. Ограничениями на *входные данные* являются максимальный размер строковых данных, диапазоны числовых типов полей структуры и допустимый размер области динамической памяти в языке С++.

Теоретические сведения.

Ознакомьтесь с материалом лекции.

Задания к лабораторной работе.

Выполните приведенные ниже задания.

1. На основе кодов функций, представленных в лекции 32, реализуйте все основные операции над красно-черным деревом.
2. Разработайте программу, с помощью которой можно определить наибольший допустимый размер *дека* с вещественным информационным полем. Найдите этот размер (число элементов в деке). Сравните с наибольшим допустимым размером стека и очереди с аналогичным информационным полем.
3. Элементами *дека* являются натуральные числа. Удалите из *дека* элементы, оставив только простые числа. Расположите их в порядке неубывания.
4. Удалите из циклического однонаправленного списка все отрицательные числа.
5. Решите задачу Иосифа Флавия с помощью циклического списка.
6. В красно-черном дереве найдите путь от корня к некоторому листу, содержащий минимальное количество красных вершин.

Указания к выполнению работы.

Выполнение работы следует начать с решения задачи 1, реализовав алгоритмы основных операций над красно-черным деревом. Каждое из заданий необходимо решить в соответствии с изученными методами и реализованными алгоритмами формирования, вывода и обработки данных динамических структур в языке С++. Обработку динамических структур следует выполнить на основе базовых алгоритмов: *поиск* структуре, вставка элемента в структуру, балансировка красно-черного дерева, *удаление элемента* из структуры, удаление всей динамической структуры. При объявлении динамических структур выполните комментирование используемых полей. Задача 2 носит исследовательский характер, поэтому следует отразить в отчете подробное описание предлагаемого метода определения максимального размера *дека* и выводы по сравнению с размерами стека и очереди. Программу для решения каждого задания необходимо разработать методом процедурной абстракции, оформив комментарии к коду.

Следует реализовать каждое задание в соответствии с приведенными этапами:

- изучить словесную постановку задачи, выделив при этом все виды данных;
- сформулировать математическую постановку задачи;
- выбрать метод решения задачи, если это необходимо;
- разработать графическую схему алгоритма;
- записать разработанный алгоритм на языке С++;
- разработать контрольный тест к программе;
- отладить программу;
- представить отчет по работе.

Требования к отчету.

Отчет по лабораторной работе должен соответствовать следующей структуре.

- Титульный лист.
- Словесная постановка задачи. В этом подразделе проводится полное описание задачи. Описывается суть задачи, анализ входящих в нее физических величин, область их допустимых значений, единицы их измерения, возможные ограничения, анализ условий при которых задача имеет решение (не имеет решения), анализ ожидаемых результатов.
- Математическая модель. В этом подразделе вводятся математические описания физических величин и математическое описание их взаимодействий. Цель подраздела – представить решаемую задачу в математической формулировке.
- Алгоритм решения задачи. В подразделе описывается разработка структуры алгоритма, обосновывается абстракция данных, задача разбивается на подзадачи. Схема алгоритма выполняется по ЕСПД (ГОСТ 19.003-80 и ГОСТ 19.002-80).
- Листинг программы. Подраздел должен содержать текст программы на языке программирования С++, реализованный в среде MS Visual Studio 2010.
- Контрольный тест. Подраздел содержит наборы исходных данных и полученные в ходе выполнения программы результаты.
- Выводы по лабораторной работе.
- Ответы на контрольные вопросы.

Тема 10: «Рекурсия и рекурсивные алгоритмы»

Лабораторная работа: Рекурсия и рекурсивные алгоритмы

Цель работы: изучить понятие, виды рекурсии и рекурсивную триаду, научиться разрабатывать рекурсивную триаду при решении задач на языке С++.

При выполнении лабораторной работы для каждого задания требуется написать программу на языке С++, которая получает на входе числовые данные, выполняет их обработку в соответствии с требованиями задания и выводит результат на экран. Для обработки данных необходимо реализовать *рекурсивную функцию*. Ввод данных осуществляется с клавиатуры с учетом требований к входным данным, содержащихся в постановке задачи (ввод данных сопровождайте диалогом). Ограничениями на *входные данные* является допустимый *диапазон* значений используемых числовых типов в языке С++.

Теоретические сведения.

Ознакомьтесь с материалом лекции.

Задания к лабораторной работе.

Выполните приведенные ниже задания.

1. Разработайте *рекурсивную функцию*, подсчитывающую количество способов разбиения выпуклого многоугольника на треугольники непересекающимися диагоналями.
2. В Фибоначчиевой системе *счисления* числа формируются по правилам.
   - Используются только символы 0 и 1;
   - Каждый разряд соответствует элементу последовательности Фибоначчи 1, 2, 3, 5, 8, …, то есть указывает на наличие или отсутствие такового;
   - В соседних разрядах не могут стоять символы 1, так как это автоматически означает формирование следующего за ними разряда. Например, $17_{10} = 13_{10} + 3_{10} + 1_{10} = 100101_{ф}$.

     Составьте программу перевода числа из десятичной системы в Фибоначчиевую. Считать входные данные введенными корректно.
3. Найдите походящие дроби рационального числа x/y (x – неотрицательно, y – положительно). Например, $\frac{5}{6} = 0 + \frac{1}{1+\frac{1}{5}}$, то есть для x = 5, y = 6 ответом будет последовательность [0; 1, 5].
4. Вычислите определитель квадратной матрицы размера nxn.

Указания к выполнению работы.

Каждое задание необходимо решить в соответствии с изученными рекурсивными методами решения задач и методами обработки числовых данных в языке C++. Перед реализацией кода каждой задачи необходимо разработать рекурсивную триаду в соответствии с постановкой задачи: выполнить параметризацию, выделить базу и оформить декомпозицию рекурсии. Этапы рекурсивной триады необходимо отразить в математической модели к отчету, выполнив обоснование *декомпозиции*. Программу для решения каждого задания необходимо разработать методом процедурной абстракции, используя рекурсивные функции. Этапы сопроводить комментариями в коде.

Следует реализовать каждое задание в соответствии с приведенными этапами:

- изучить словесную постановку задачи, выделив при этом все виды данных;
- сформулировать математическую постановку задачи;
- выбрать метод решения задачи, если это необходимо;
- разработать графическую схему алгоритма;
- записать разработанный алгоритм на языке C++;
- разработать контрольный тест к программе;
- отладить программу;
- представить отчет по работе.

Требования к отчету.

Отчет по лабораторной работе должен соответствовать следующей структуре.

- Титульный лист.
- Словесная постановка задачи. В этом подразделе проводится полное описание задачи. Описывается суть задачи, анализ входящих в нее физических величин, область их допустимых значений, единицы их измерения, возможные ограничения, анализ условий при которых задача имеет решение (не имеет решения), анализ ожидаемых результатов.
- Математическая модель. В этом подразделе вводятся математические описания физических величин и математическое описание их взаимодействий. Цель подраздела – представить решаемую задачу в математической формулировке.
- Алгоритм решения задачи. В подразделе описывается разработка структуры алгоритма, обосновывается абстракция данных, задача разбивается на подзадачи. Схема алгоритма выполняется по ЕСПД (ГОСТ 19.003-80 и ГОСТ 19.002-80).
- Листинг программы. Подраздел должен содержать текст программы на языке программирования C++, реализованный в среде MS Visual Studio 2010.
- Контрольный тест. Подраздел содержит наборы исходных данных и полученные в ходе выполнения программы результаты.

- Выводы по лабораторной работе.
- Ответы на контрольные вопросы.

Контрольные вопросы

1. Можно ли случай *косвенной рекурсии* свести к прямой рекурсии? Ответ обоснуйте.
2. Может ли рекурсивная база содержать несколько тривиальных случаев? Ответ обоснуйте.
3. Являются ли параметры, база и *декомпозиция* единственными для конкретной задачи? Ответ обоснуйте.
4. С какой целью в задачах происходит пересмотр или корректировка выбранных параметров, выделенной базы или случая *декомпозиции*?
5. Является ли рекурсия универсальным способом решения задач? Ответ обоснуйте.
6. Почему для оценки трудоемкости рекурсивного алгоритма недостаточно одного метода подсчета вершин рекурсивного дерева?
7. Выполните оценку алгоритма из *Примера 3* лекции 34 методом подсчета вершин рекурсивного дерева для случая n = 6, k = 6.

Лабораторная работа. Решение задач на использование рекурсивных алгоритмов

Цель работы: изучить рекурсивные алгоритмы и основные схемы решения задач рекурсивными способами, научиться применять рекурсивные алгоритмы при решении задач на языке C++.

При выполнении лабораторной работы для каждого задания требуется написать программу на языке C++, которая получает на входе числовые данные, выполняет их обработку в соответствии с требованиями задания и выводит результат на экран. Для обработки данных необходимо реализовать *рекурсивную функцию*. Ввод данных осуществляется с клавиатуры с учетом требований к входным данным, содержащихся в постановке задачи (ввод данных сопровождайте диалогом). Ограничениями на *входные данные* является допустимый *диапазон* значений используемых *числовых типов* в языке C++.

Теоретические сведения.

Ознакомьтесь с материалом лекции.

Задания к лабораторной работе.

Выполните приведенные ниже задания.

1. Два многочлена заданы своими степенями и коэффициентами. Выполните умножение данных многочленов. Выведите в файл коэффициенты результата в порядке убывания степеней его одночленов.
2. Найдите сумму факториалов первых n натуральных чисел. Решите двумя способами: через непосредственное вычисление факториалов и с помощью преобразования декомпозиционных отношений. Оцените трудоемкость функции в каждом случае.
3. Для данных натуральных n и m найдите цепную дробь, соответствующую отношению n/m.
4. Вычислите значение функции Аккермана двумя способами: непосредственно из определения и снизив трудоемкость алгоритма. Найдите каждым способом Akkerman(3, 7) и Akkerman(8, 20). Функция Аккермана определяется рекурсивно для неотрицательных целых чисел m и n следующим образом:

$$A(m,n) = \begin{cases} n+1, \text{ при } m = 0 \\ A(m-1,1), \text{ при } m > 0, n = 0; \\ A(m-1, A(m,n-1)), \text{ при } m > 0, n > 0. \end{cases}$$

5. Первый член последовательности равен натуральному числу, сравнимому с 2 по модулю 3. Каждый следующий член последовательности равен сумме кубов цифр предыдущего члена. Исследуйте последовательности на сходимость для конкретного первого члена.

Указания к выполнению работы.

Каждое задание необходимо решить в соответствии с изученными *рекурсивными методами* решения задач и методами обработки числовых данных в языке С++. Перед реализацией кода каждой задачи необходимо разработать рекурсивную триаду в соответствии с постановкой задачи: выполнить параметризацию, выделить базу и оформить декомпозицию рекурсии. Рекомендуется воспользоваться материалами лекции 35, где подробно рассматриваются примеры разработки рекурсивной триады и обоснование декомпозиции. Этапы рекурсивной триады необходимо отразить в математической модели к отчету, выполнив обоснование декомпозиции. Программу для решения каждого задания необходимо разработать методом процедурной абстракции, используя рекурсивные функции. Этапы сопроводить комментариями в коде. В отчете следует отразить результаты тестирования программ.

Следует реализовать каждое задание в соответствии с приведенными этапами:

- изучить словесную постановку задачи, выделив при этом все виды данных;
- сформулировать математическую постановку задачи;
- выбрать метод решения задачи, если это необходимо;
- разработать графическую схему алгоритма;
- записать разработанный алгоритм на языке С++;
- разработать контрольный тест к программе;
- отладить программу;
- представить отчет по работе.

Требования к отчету.

Отчет *по* лабораторной работе должен соответствовать следующей структуре.

- Титульный лист.
- Словесная постановка задачи. В этом подразделе проводится полное описание задачи. Описывается суть задачи, анализ входящих в нее физических величин, область их допустимых значений, единицы их измерения, возможные ограничения, анализ условий при которых задача имеет решение (не имеет решения), анализ ожидаемых результатов.
- Математическая модель. В этом подразделе вводятся математические описания физических величин и математическое описание их взаимодействий. Цель подраздела – представить решаемую задачу в математической формулировке.
- Алгоритм решения задачи. В подразделе описывается разработка структуры алгоритма, обосновывается абстракция данных, задача разбивается на подзадачи. Схема алгоритма выполняется по ЕСПД (ГОСТ 19.003-80 и ГОСТ 19.002-80).
- Листинг программы. Подраздел должен содержать текст программы на языке программирования С++, реализованный в среде MS Visual Studio 2010.

- Контрольный тест. Подраздел содержит наборы исходных данных и полученные в ходе выполнения программы результаты.
- Выводы по лабораторной работе.
- Ответы на контрольные вопросы.

Контрольные вопросы

1. Почему рекурсию нельзя рассматривать как универсальный метод решения задач?
2. Почему опорные схемы решения задач рекурсивными способами не являются жестко привязанными к отдельным классам задач?
3. Каким образом анализ решения обратной задачи может привести к решению поставленной задачи?
4. По каким признакам "находится родственник" в одноименной опорной схеме?
5. Всегда ли отбрасывание условий и переход к подзадаче могут привести к эквивалентной задаче? Обоснуйте ответ примерами.
6. Какие свойства объектов выступают в роли характеристических в соответствующей опорной схеме?

Задачи:

1. Задача о разрезании прямоугольника на квадраты.

Дан *прямоугольник*, стороны которого выражены натуральными числами. Разрежьте его на минимальное число квадратов с натуральными сторонами. Найдите число получившихся квадратов.

Разработаем рекурсивную триаду.

Параметризация: m, n – *натуральные числа*, соответствующие размерам прямоугольника.

База рекурсии: для m=n число получившихся квадратов равно 1, так как данный *прямоугольник* уже является квадратом.

Декомпозиция: если $m \neq n$, то возможны два случая m < n или m > n. Отрежем от прямоугольника наибольший по площади квадрат с натуральными сторонами. *Длина* стороны такого квадрата равна наименьшей из сторон прямоугольника. После того, как квадрат будет отрезан, размеры прямоугольника станут следующие: большая сторона уменьшится на длину стороны квадрата, а меньшая не изменится. Число искомых квадратов будет вычисляться как число квадратов, на которые будет разрезан полученный *прямоугольник*, плюс один (отрезанный квадрат). К получившемуся прямоугольнику применим аналогичные рассуждения: проверим на соответствие базе или перейдем к *декомпозиции* (рис. 1).

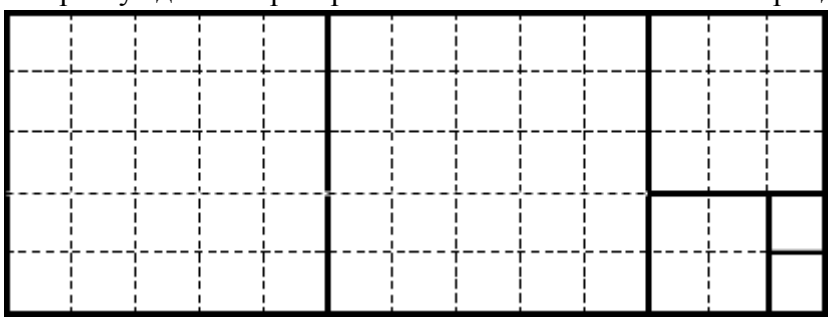


Рис. 1. Пример разрезания прямоугольника 13x5 на квадраты

2. Задача о нахождении центра тяжести выпуклого многоугольника.

Выпуклый многоугольник задан на плоскости координатами своих вершин. Найдите его центр тяжести.
Разработаем рекурсивную триаду.
Параметризация: x, y – вещественные массивы, в которых хранятся *координаты* вершин многоугольника; n – это число вершин многоугольника, по условию задачи, n>1 так как минимальное число вершин имеет двуугольник (*отрезок*).
База рекурсии: для n=2 в качестве многоугольника рассматривается *отрезок*, центром тяжести которого является его середина (рис. 34A). При этом середина делит *отрезок* в отношении 1 : 1. Если *координаты* концов отрезка заданы как (x_0,y_0) и (x_1,y_1),
то *координаты* середины вычисляются по формуле:

$$cx = \frac{x_0 + x_1}{2}, \quad cy = \frac{y_0 + y_1}{2}.$$

Декомпозиция: если n>2, то рассмотрим последовательное нахождение центров тяжести треугольника, четырехугольника и т.д.
Для n=3 центром тяжести треугольника является точка пересечения его медиан, которая делит каждую медиану в отношении 2 : 1, считая от вершины. Но *основание медианы* – это середина отрезка, являющегося стороной треугольника. Таким образом, для нахождения центра тяжести треугольника необходимо: найти центр тяжести стороны треугольника (отрезка), затем разделить в отношении 2 : 1, считая от вершины, *отрезок*, образованный основанием *медианы* и третьей вершиной (рис. 34B).
Для n=4 центром тяжести четырехугольника является точка, делящая в отношении 3 : 1, считая от вершины, *отрезок*: он образован центром тяжести треугольника, построенного на трех вершинах, и четвертой вершиной (рис. 2).

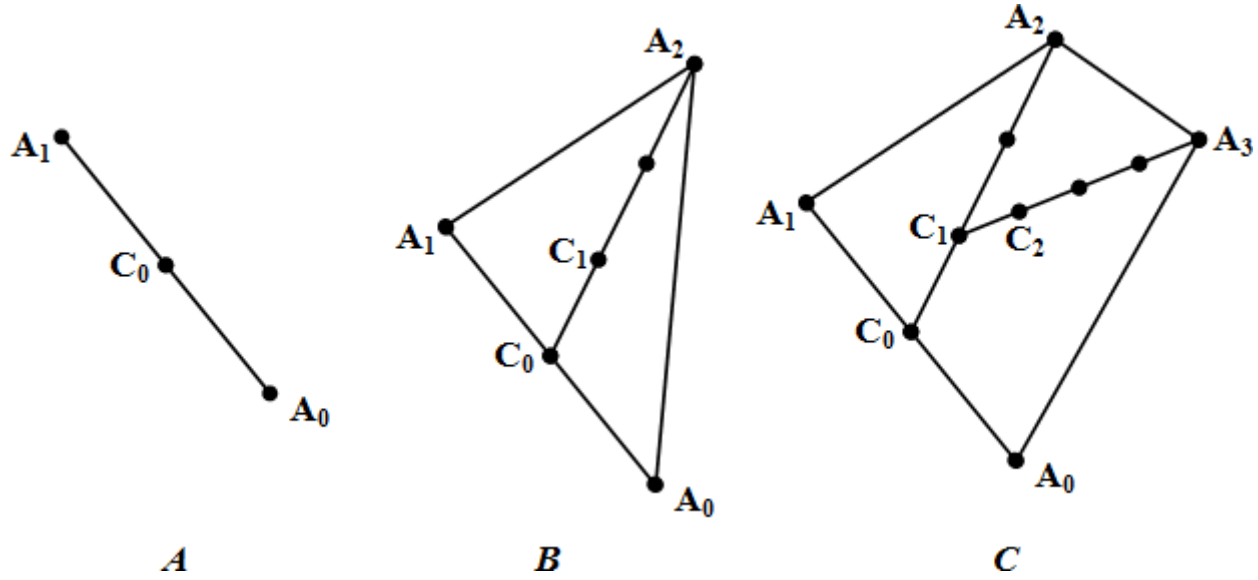


Рис. 2. Примеры построения центров тяжести многоугольников

Таким образом, для нахождения центра тяжести n -угольника необходимо разделить в отношении (n-1): 1, считая от вершины, *отрезок*: он образован центром тяжести (n-1) -угольника и n -ой вершиной рассматриваемого многоугольника. Если концы отрезка заданы координатами вершины (x_n,y_n) и центра тяжести (n-1) -угольника (cx_{n-1},cy_{n-1}), то при делении отрезка в данном отношении получаем *координаты*:

$$cx_n = \frac{x_n + (n-1)cx_{n-1}}{n}, \quad cy_n = \frac{y_n + (n-1)cy_{n-1}}{n}$$

Тема 11: «Основные принципы объектно-ориентированного программирования».

Контрольные вопросы:

1. В чем основные отличия класса от структуры?
2. Дайте понятие наследования классов.
3. Опишите класс для хранения имени, места работы и возраста сотрудника с двумя конструкторами: без аргументов и с аргументами для инициализации указанных полей.
4. Какими способами можно создавать экземпляры классов?
5. Дайте понятие полиморфизма.
6. В какой последовательности вызываются конструкторы базовых классов при создании экземпляра дочернего класса?
7. При каком режиме доступа возможно обращение ко всем элементам класса?
8. Придумайте и запишите какой-либо метод класса для задания значений его частным элементам.
9. Каким образом выполняется наследование классов в C++?
10. Как задается описание функции класса за его пределами?
11. В чем особенность режима доступа protected и чем он отличается от режима private?
12. Дайте понятие множественного наследования.
13. Что такое дружественные функции и для чего они предназначены?
14. Как задаются виртуальные функции класса?
15. Запишите двухуровневую иерархию для описания объема хранимых денежных средств в разной валюте и в базовом классе реализуйте виртуальную функцию для вывода доступных средств в соответствующих денежных единицах.
16. Поясните, что понимается под перегрузкой операторов.
17. Запишите класс для работы с комплексными числами, используя механизм перегрузки операторов.

Задачи:

1. Создание класса и объектов типа "полином" с динамическим выделением памяти под элементы, для которого определены основные операции: сложение, вычитание, присваивание, *умножение, *деление. Реализация различных типов конструкторов для одного типа, включая конструктор копирования. Использование переопределенных операторов потокового ввода-вывода для этого нового типа. Иллюстрация на простом примере использования объектов нового типа.
2. Создание нескольких (не менее двух) типов для графических объектов, используя механизм наследования от классов Point или Circle. (например: квадрат, прямоугольник, возможно закрашенный, круг, сектор, *3D- объекты и т.п.) Иллюстрация на простом примере использования объектов нового типа.
3. Создание двухуровневой иерархии классов, где в качестве родительского класса выступает абстрактный класс, на примере абстрактного типа "график функции". При этом вид функции определяется в производном классе. Реализация операций сложения функций и умножения на константу, используя дружественные функции. Иллюстрация на простом примере использования объектов нового типа.

Тема 12: «Объектно-ориентированная модель. Этапы разработки программных продуктов с использованием ООП».

Вопросы для устного обсуждения:

Практическая работа: Разработка программы-калькулятора дробей

Формулировка задания

Требуется разработать программу-калькулятор дробей: fc (fraction calculator), который обеспечивает выполнение арифметических операций (сложение, вычитание, умножение и деление) для рациональных дробей, заданных в символическом формате. Программа fc должна быть ориентированна на использование следующего формата дробей: a/b, где a и b наборы цифр, отображающие числитель и знаменатель дроби.

Исходными данными программы fc являются последовательности символов, представляющие вычисляемые арифметические выражения, где дробные операнды связывает символ операции(+, -, *, /). Дробные операнды и символ операции должны передаваться программе fc, через параметры командной строки её вызова. В командной строке операнды и символ операции разделяются с помощью пробелов.

Результатом работы программы fc должно быть дробное число, которое представляет итог вычисления входного выражения и отображается в потоке стандартного вывода (stdout). Для некоторых операций и операндов должен быть предусмотрен вывод соответствующих информационных сообщений в поток протокола стандартной диагностики (stderr).

Программа fc должна быть составлена в системе программирования C++ на основе разработки специального класса, в котором перегружены специальные арифметические операции над дробными числами (выполнена перегрузка операций).

Свойства дробных чисел

В некоторых алгоритмах численной обработки данных более полезным является точное выражение результата в виде дроби (например, 1/3), чем приближенное представление с плавающей точкой (0,33333...). Использование дробей позволяет получить нецелочисленный ответ задачи в наглядной форме и исключить ошибку округления, свойственные обработке чисел с плавающей точкой в ограниченной разрядной сетке. Как известно, дробное число образует отношение 2-х целых чисел, которое записывается следующем образом: a/b, где a и b - наборы цифр, представляющие числитель и знаменатель дроби. Относительно числителя и знаменателя дроби приняты следующие договоренности:

- знаменатель дроби должен быть натуральным числом (b>0);
- целое число можно представить дробью с единичным знаменателем (a/1);
- знак дроби определяется знаком числителя;
- если числитель и знаменатель дроби умножить на одно и тоже натуральное число (N), то получиться дробь равная данной (a/b=(N*a)/(N*b));
- если числитель и знаменатель дроби имеют общий делитель, то при делении их на него происходит сокращение дроби и образуется дробь равная данной (D*a)/(D*b)=a/b);
- если числитель и знаменатель взаимно-простые числа, т. е. не имеют общих делителей, то дробь называется несократимой;
- любая дробь может быть представлена к несократимой, если её числитель сократить на их наибольший общий делитель (Hog) - наибольшее натуральное число, на которое они оба делятся без остатка;
- две любые дроби a/b и c/b считаются равными, если (a*d)=(b*c);
- две несократимые дроби считаются равными, если равны их числители и знаменатели (a=c и b=d).

Вычислительная обработка дробных чисел основана на использовании 4-х арифметических операций: сложение, вычитание, умножение и деление. Для выполнения этих

операций символическое представление дроби выражается парой целых чисел, соответствующих её числителю и знаменателю. Пусть U/U' и V/V' обозначают дроби-операнды, а W/W'-дроби, результат операции. Тогда числитель и знаменатель результирующей дроби для 4-х арифметических операций выражают следующие соотношения:

Умножение:

(W/W')={(U/U')*(V/V')}

W=(U/d1)*(V/d2) и W'=(U'/d2)*(V'/d1), где d1=Hog(U,V') и d2=Hog(U'/V).

Деление: (W/W')={(U/U')/(V/V')}

W=(U/d1)*(V'/d2)sign(V) и W'=|(U'/d2)*(V/d1)|, где d1=Hog(U,V) и d2=Hog(U',V').

Сложение: (W/W')={(U/U')+(V/V')}

Если Hog(U'/V')=1,то

W=(U*V')+(U'*V) и W'=(U'*V')

Если Hog(U'/V')>1,то

W=t/d2 и W'=(U'/d1)*(V'/d2), где d1=Hog(U',V'), t=U*(V'/d2)+V*(U'/d1) и d2=Hog(t,d1).

Вычитание: (W/W')={(U/U')-(V/V')}. Выполняется аналогично сложению, если везде заменить знак плюс на знак минус.

В приведенных соотношениях Hog(m,n) обозначает наибольший общий делитель чисел |m| и |n|. Для эффективного вычисления Hog применяется алгоритм Евклида - самый старый нетривиальный алгоритм, доживший до наших дней. В современной редакции псевдокод алгоритма Евклида выглядит следующим образом:

Пример 1

```
IF m<0 THEN
m <- |m|;  /* перейти к абсолютной величине числа m */
IF n<0 THEN
n <- |n|;  /* перейти к абсолютной величине числа n */
WHILE n<>0 { /* цикл уменьшения n */
r <- m mod n; /* остаток деления m на n */
m <- n;
n <- r;
}
RETURN m.
```

В классическом варианте алгоритм Евклида используется для поиска наибольшего общего делителя пары не отрицательных целых чисел, при условии, что Hog(0,0) принимается равным 0. Для расширения области применения на поле произвольных целых чисел в предлагаемой версии алгоритма Евклида предусмотрен переход к абсолютным величинам рассматриваемых чисел. На каждом шаге основного цикла алгоритма Евклида происходит последовательное уменьшение обоих чисел, но при этом значение их наибольшего общего делителя остается неизменным, т. к.:

Цикл уменьшения обоих чисел продолжается, пока на очередной итерации меньшее из чисел не станет равным 0. В этом случае большее из чисел принимается за наибольший общий делитель исходной пары, т. к.:

Пример 2

Hog(259,111) выполняется за 3 шага: Hog(259,111)=Hog(111,37)=Hog(3,0)=3.

Сложность алгоритма Евклида пропорциональна логарифму от максимального по абсолютной величине числа из рассматриваемой пары чисел. эффективность алгоритма Евклида определяет эффективность выполнения арифметических операций над дробями по соотношениям, определенным выше.

На первый взгляд кажется, что арифметические операции над дробями можно реализовать более эффективно, вычисляя наибольший общий делитель в каждой операции только один раз вместо двух. Например, при умножении {(U/U')*(V/V')}, числитель и знаменатель ответа можно получить по внешне более простым формулам:

W=(U*V)/d и W'=(U/U')/d,

где $d=Hog(U*V,U'*V')$

При сложении 2-х дробей {(U/U')+(V/V')} можно представить результат в виде следующей дроби:

$(U*V'+V*U')/(U'*V')$

а затем привести результирующую дробь к не сократимому виду, используя свойство: $Hog(U*V'+V*U',U'*V')$.

Однако, в обоих случаях придется оперировать с относительно большими числами на каждой итерации алгоритма Евклида. Каждая итерация будет выполнятся медленнее, чем код на в ней рассматриваются меньшие числа, которые характерны для операций обработки дробей с 2-мя вычислениями наибольшего общего делителя. Таким образом, по быстродействию оба подхода по-крайней мере равнозначны, но в первом исключена возможность переполнения разрядной сетки, что является решающим преимуществом, когда числители и знаменатели дробей являются большими числами.

Программирование калькулятора дробей.

При разработке программы калькулятора дробей целесообразно сосредоточить описание дробных структур данных и операции их обработки в отдельном классе Fraction. Класс Fraction должен содержать частные (private) компоненты данные и общедоступные (public) компонентные методы обработки приватных данных. Это позволит оперировать данными класса Fraction только в компонентных методах и исключит возможность непосредственного обращения к ним из любой внешней функции программы. Исходя из этого декларация логической структуры класса Fraction должна иметь следующий формат:

```
class Fraction {
private: /* спецификация компонентных данных */
public:  /* объявление прототипов компонентных методов */
};
```

В приватную часть класса Fraction следует включить спецификацию целочисленных компонентных данных, которые обозначают числитель и знаменатель дроби, как показано в следующем фрагменте класса Fraction:

```
class Fraction {
private:
int nominator; /* числитель */
int denominator; /* знаменатель */
public:
.....
};
```

В общедоступную часть декларации класса Fracton нужно включить объявление прототипов компонентных методов, которые реализуют арифметическую обработку дробей и отображение результатов обработки. Кроме того, в общедоступной части следует объявить конструктор класса.

Желаемую арифметическую обработку дробных чисел удобно реализовать с помощью компонентных оператор-функций, которые обеспечат перегрузку арифметических операций (+,-,*,/) в выражениях с объектами класса Fraction. Наличие оператор-функций позволит конструировать арифметические выражения для объектов класса Fraction в наглядной форме как для объектов простых типов. При наличии соответствующих оператор-функций будет правильно вычисляться, например, следующее выражение программы:

```
r=f1@f2;
```

где r,f1 и f2 объекты класса Fraction, а @ - знак операции.

По формату определения и декларации оператор-функция идентична обычной функции с предопределенным именем operator@, где @ обозначает знак перегружаемой операции, и специальным соглашением по аргументам. Для декларации 4-х требуемых оператор-функций в классе Fraction может быть рекомендован следующий идентичный формат:

```
Fraction operator@( Fraction& );
```

который отличается только символом перегружаемой операции @ (+,-,*,/). Используя эти декларации, компилятор языка C++ будет рассматривать приведенное выше выражение как следующий вызов соответствующей компонентной оператор-функции:

```
r=f1 operator@(f2);
```

Как видно из приведённой спецификации вызова, аргумент компонентной оператор-функции используется для передачи по ссылке 2-го операнда операции. Первый операнд операции неявно передается через скрытый аргумент, присущий любой компонентной функции и доступный по указателю this. Результат вычислений в теле оператор-функции возвращается в основную программу для присвоения объекту класса Fraction. Следует отметить, что передача аргумента по ссылке в данном случае выбрана из экономических соображений. Для оператор-функции допустима передача аргумента по значению. Однако, в этом случае происходит создание промежуточной копии объекта передачи в стеке оператор-функции.

Кроме компонентных оператор-функций в состав компонентных методов класса Fraction целесообразно включить 3 обычные компонентные функции:

```
int euclid(int,int);
int sign(int);
void ptint(void);
```

декларировав их в общедоступной части класса Fraction, указанным образом.

Компонентный метод euclid должен вычислять наибольший общий делитель своих аргументов, используя алгоритм Евклида. Вычисленное значение должно передаваться через код возврат метода. Этот метод предназначается для реализации алгоритмов арифметических операций с дробями в компонентных оператор-функциях класса Fraction.

Компонентный метод sign используется для определения знаков своего аргумента, возвращая (-1), если значение аргумента меньше 0 или +1, если аргумент имеет неотрицательное значение. Этот метод следует использовать для реализации алгоритма деления дробных чисел в соответствующей компонентной оператор-функции класса Fraction.

Компонентный метод print должен отображать в потоке стандартного вывода числитель и знаменатель объекта класса Fraction, от имени которого он вызван. Этот метод следует применять в программе калькулятора дробей для отображения результата операций с дробями.

Для инициализации компонентных данных при создании объектов класса Fraction в нем необходимо предусмотреть конструктор с аргументом типа указатель на строку символов (char*). Поэтому декларация конструктора в общественной части объявления класса Fraction должен иметь следующий формат:

```
Fraction(char*);
```

Через аргумент в этот конструктор должно передаваться представление дроби в виде строки символов, где числитель и знаменатель выражаются выборами цифр, которые

разделяет символ '/'. Конструктор должен преобразовывать символьное представление числителя и знаменателя в целочисленные значения компонент-данных класса Fraction, используя библиотечную функцию atoi системы программирования C++.

Рассмотренные общедоступные компоненты класса Fraction можно использовать в основной функции main(int argc,char** argv) программы калькулятора дробей fc. Вычисляемое арифметическое выражение должно передаваться в функцию main через параметры командной строки вызова программы fc, так что 1-й (argv[1]) и 3-й

(argv[3]) параметры соответствуют дробным операндам.

В теле функции main следует задать 3 объекта класса Fraction для хранения операндов и результата операции. Объекты- операнды должны быть инициализированы строками соответствующих параметров командной стоки. Объект результата можно инициализировать произвольной дробью, представленной в символическом формате, например, "0/1".

Знак операции, передаваемый в функцию main через 2-й параметр командной строки, удобно рассматривать в операторе switch системы программирования C++ для выбора одного из 4-х вариантов вычисляемых дробных выражений. После вычисления выбранного выражения с помощью соответствующей компонентной оператор- функции класса Fraction ответ нужно сохранить в зарезервированном объекте-результате. Вызов компонентной функции print класса Fraction от имени объекта результата, позволит отобразить полученную дробь в потоке стандартного вывода.

Контрольные задания

1. Расширить класс Fraction перегрузкой операции присваивания (=), в которой должно быть реализовано копирование числителя и знаменателя, а также приведение результирующей дроби к несократимому виду, если это необходимо.

2. Расширить класс Fraction перегрузкой операции проверки равенства 2-х дробей (==).

3. Модифицировать класс Fraction, так чтобы перегрузку операций над его объектами обеспечивали дружественные оператор-функции. При этом необходимо учесть, что прототипы дружественных функций должны быть объявлены в декларации класса с модификатором friend и в них по ссылке должны передаваться оба операнда, т.к. скрытый указатель this на объект класса в них недоступен.

4. Усовершенствовать представление результата операций с дробями, так чтобы в случае, когда числитель больше знаменателя, выделялась целая часть числа.

5. Разработать средства контроля достоверности результатов операций с дробными числами. Например, сложение дробей должно проверяться вычитанием, а умножение - делением.

6. Переориентировать программу-калькулятор для интерактивного ввода вычисляемых арифметических выражений с дробями из потока стандартного ввода.

7. Реализовать операции сложения и вычитания с помощью приведения дробей к общему знаменателю. Для вычисления наименьшего общего кратного (Hok) знаменателей дробей, которое необходимо в этом случае, рекомендуется использовать следующее соотношение:

U'*V'=Hog(U',V')*Hok(U',V').

Тема 13: «Алгоритмы на графах. Алгоритмы обхода графа».

Лабораторная работа: Алгоритмы на графах. Алгоритмы обхода графа

Цель работы: изучить основные алгоритмы *обхода графа* и научиться решать задачи *обхода графа* на основе *поиска в ширину* и поиска в глубину.

При выполнении лабораторной работы для каждого задания требуется написать программу на языке C++, которая получает на входе числовые данные, выполняет их обработку в

соответствии с требованиями задания и выводит результат на экран. Для обработки данных необходимо реализовать алгоритмы *обхода графа* в соответствии с постановкой задачи. Ввод данных осуществляется из файла с учетом требований к входным данным, содержащихся в постановке задачи. Ограничениями на *входные данные* является допустимый *диапазон* значений используемых числовых типов в языке C++.

Теоретические сведения.

Ознакомьтесь с материалом лекции.

Задания к лабораторной работе.

Выполните приведенные ниже задания.

1. На основании приведенной в лекции 44 функции реализуйте программу, в которой выполняется алгоритм *обхода графа* на основе поиска в глубину.
2. На основании приведенной в лекции 44 функции реализуйте программу, в которой выполняется алгоритм *обхода графа* на основе *поиска в ширину*.
3. Используйте *обход графа* в ширину для определения всех *вершин графа*, находящихся на фиксированном расстоянии d от данной вершины.
4. Перенумеруйте вершины графа в порядке *обхода в глубину* и вычислите среднюю плотность графа как частное от деления количества его ребер на число вершин. Можно ли оба эти действия выполнить за один *обход графа*?
5. В вершинах неориентированного графа хранятся положительные целые числа. Подсчитайте количество пар дружественных чисел в вершинах графа, которые соединены ребрами.

Указания к выполнению работы.

Каждое задание необходимо решить в соответствии с изученными алгоритмами *обхода графа*, реализовав программный код на языке C++. Рекомендуется воспользоваться материалами лекции 44, где подробно рассматриваются описания алгоритмов *обхода графа*, примеры разработки функций, реализующих алгоритмы *обхода графа*, на языке C++. Программу для решения каждого задания необходимо разработать методом процедурной абстракции, используя функции. Этапы решения сопроводить комментариями в коде. В отчете следует отразить разработку и обоснование математической модели решения задачи, представить результаты тестирования программ.

Следует реализовать каждое задание в соответствии с приведенными этапами:

- изучить словесную постановку задачи, выделив при этом все виды данных;
- сформулировать математическую постановку задачи;
- выбрать метод решения задачи, если это необходимо;
- разработать графическую схему алгоритма;
- записать разработанный алгоритм на языке C++;
- разработать контрольный тест к программе;
- отладить программу;
- представить отчет по работе.

Требования к отчету.

Отчет по лабораторной работе должен соответствовать следующей структуре.

- Титульный лист.
- Словесная постановка задачи. В этом подразделе проводится полное описание задачи. Описывается суть задачи, анализ входящих в нее физических величин, область их допустимых значений, единицы их измерения, возможные ограничения,

анализ условий при которых задача имеет решение (не имеет решения), анализ ожидаемых результатов.

- Математическая модель. В этом подразделе вводятся математические описания физических величин и математическое описание их взаимодействий. Цель подраздела – представить решаемую задачу в математической формулировке.
- Алгоритм решения задачи. В подразделе описывается разработка структуры алгоритма, обосновывается абстракция данных, задача разбивается на подзадачи. Схема алгоритма выполняется по ЕСПД (ГОСТ 19.003-80 и ГОСТ 19.002-80).
- Листинг программы. Подраздел должен содержать текст программы на языке программирования C++, реализованный в среде MS Visual Studio 2010.
- Контрольный тест. Подраздел содержит наборы исходных данных и полученные в ходе выполнения программы результаты.
- Выводы по лабораторной работе.
- Ответы на контрольные вопросы.

Контрольные вопросы

1. Как связаны между собой различные способы представления графов?
2. Как от вида или представления графа зависит временная сложность алгоритмов поиска в глубину и в ширину?
3. Как при реализации в коде выполняется возвращение из тупиковых вершин при обходе графа?
4. Как выполняется обход в несвязном графе?
5. Распространяются ли понятия "*поиск в глубину*" и "*поиск в ширину*" на несвязный граф? Ответ обоснуйте.
6. Охарактеризуйте трудоемкость рекурсивного и нерекурсивного алгоритмов *обхода графа*.

Тема 14: «Методы проектирования программ».

Задание.

Решите уравнение указанным в варианте методом. Функцию передать как *параметр* с помощью указателя.

Численные методы решения уравнений

Довольно часто на практике приходится решать уравнения вида:

$$F(x) = 0 \quad (2)$$

где функция $F(x)$ определена и непрерывна на некотором конечном или бесконечном интервале

$$\alpha < x < \beta$$

Всякое значение $\overline{x}$ такое, что $F(\overline{x}) \equiv 0$, называется корнем уравнения, а нахождение этого значения и есть решение уравнения.

На практике в большинстве случаев найти точное решение возникшей математической задачи не удается. Поэтому важное значение приобрели численные методы, позволяющие найти приближенное значение корня. Под численными методами подразумеваются методы решения задач, сводящиеся к арифметическим и некоторым логическим действиям над числами, т.е. к тем действиям, которые выполняет компьютер.

Существует множество *численных методов решения уравнений* вида (1). Рассмотрим только три из них:

- *метод итераций*;
- *метод Ньютона*;
- *метод половинного деления*.

Метод итераций

Представим уравнение $F(x) = 0$ в виде:

$x = f(x)$ (2)

Это уравнение получается выделением x из уравнения $F(x)$ и переносом того, что осталось, т.е. $f(x)$, в левую часть уравнения. Иначе можно получить уравнение (2) следующим способом: левую и правую часть уравнения (1) умножить на произвольную константу λ и прибавить к левой и правой части x, т.е. получаем уравнение вида:

$x = x + \lambda F(x)$ (3)

где $f(x) = x + \lambda F(x)$.

На заданном отрезке $[a; b]$ выберем точку x_0 – нулевое приближение – и найдем

$x_1 = f(x_0),$

потом найдем:

$x_2 = f(x_1),$

Таким образом, процесс нахождения корня уравнения сводится к последовательному вычислению чисел:

$x_n = f(x_{n-1}) \quad n = 1, 2, 3 \ldots .$

Этот процесс называется методом итераций.

Если на отрезке $[a; b]$ выполнено условие:

$|f'(x_0)| \le q < 1,$

то процесс итераций сходится, т.е.

$\lim\limits_{n\to\infty} x_n = \overline{x}$

Процесс итераций продолжается до тех пор, пока

$|x_n - x_{n-1}| \le \varepsilon,$

где – ε заданная *абсолютная погрешность* корня x. При этом будет выполняться:

$|\overline{x} - x_n| \le \varepsilon,$

Метод Ньютона

Пусть уравнение $F(x) = 0$ имеет один корень на отрезке $[a; b]$, причем $F'(x)$ и $F''(x)$ определены, непрерывны и сохраняют постоянные знаки на отрезке $[a; b]$.

Выберем на отрезке $[a; b]$ произвольную точку x_0 – нулевое приближение. Затем найдем:

$$x_1 = x_0 - \frac{F(x_0)}{F'(x_0)}$$

потом

$$x_2 = x_1 - \frac{F(x_1)}{F'(x_1)}$$

Таким образом, процесс нахождения корня уравнения сводится к вычислению чисел x_n по формуле:

$$x_n = x_{n-1} - \frac{F(x_{n-1})}{F'(x_{n-1})}, \quad n = 1, 2, 3 \ldots$$

Этот процесс называется методом Ньютона.

Процесс вычисления продолжается до тех пор, пока не будет выполнено условие:

$$|x_n - x_{n-1}| \leq \varepsilon,$$

где – ε заданная *абсолютная погрешность* корня x.

Точку x_0 необходимо выбирать так, чтобы выполнялось условие:

$$F(x_0)\dot{F}'(x_0) > 0,$$

иначе метод не будет сходиться.

Метод половинного деления

Пусть уравнение $F(x_0)$ имеет один корень на отрезке $[a, b]$. Функция непрерывна на отрезке $[a, b]$.

Метод половинного деления заключается в следующем:

Сначала выбираем начальное приближение, деля *отрезок* пополам, т.е.

$$x_0 = (a + b)/2.$$

Если $F(x_0) = 0$, то x_0 является корнем уравнения. Если $F(x_0) \neq 0,$ то выбираем тот из отрезков, на концах которого функция имеет противоположные знаки. Полученный *отрезок* снова делим пополам и выполняем действия сначала и т.д.

Процесс деления отрезка продолжаем до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданной точности ε, т.е. пока не будет выполняться условие:

$$|x_n - x_{n-1}| \leq \varepsilon.$$

Варианты задания

№	Уравнение	*Отрезок,* содержащий корень	Метод	Значение корня с точностью 10^{-4}
1	$3\sin\sqrt{x} + 0,35x - 3,8 = 0$	[2;3]	Итераций	2,2985
2	$0,25x^3 + x - 1,2502 = 0$	[0;2]	Ньютона	1,0001
3	$x - \frac{1}{3+\sin 3,6x} = 0$	[0;0,85]	Итераций	0,2624
4	$0,1x^2 - x\ln x = 0$	[1;2]	Ньютона	1,1183
5	$\operatorname{tg} x = \frac{1}{3}\operatorname{tg}^3 x + \frac{1}{5}\operatorname{tg}^5 x - \frac{1}{3} = 0$	[0;8]	Половинного деления	0,3333
6	$\arccos x - \sqrt{1 - 0,3x^3} = 0$	[0;1]	Итераций	0,5629
7	$3x - 4\ln x - 5 = 0$	[2;4]	Ньютона	3,2300
8	$\cos\frac{2}{x} - 2\sin\frac{1}{x} + \frac{1}{x} = 0$	[1;2]	Половинного деления	1,8756

9	$\sqrt{1-0,4x^2}-\arcsin x=0$	[0;1]	Итераций	0,7672
10	$e^x-e^{-1}-2=0$	[0;1]	Ньютона	0,8814
11	$\sin(\ln x)-\cos(\ln x)+2\ln x=0$	[1;3]	Половинного деления	1,3749
12	$x-2+\sin\frac{1}{x}=0$	[1,2;2]	Итераций	1,3077
13	$e^x+\ln x-10x=0$	[3;4]	Ньютона	3,5265
14	$\cos x-e^{\frac{-x^2}{2}}+x-1=0$	[1;2]	Половинного деления	1,0804
15	$1-x+\sin x-\ln(1+x)=0$	[0;1,5]	Итераций	1,1474
16	$3x-14+e^x-e^{-x}=0$	[1;3]	Ньютона	2,0692
17	$\sqrt{1-x}-\operatorname{tg}x=0$	[0;1]	Половинного деления	0,5768
18	$x+\cos(x^{0.52}+2)=0$	[0,5;1]	Итераций	0,9892
19	$3\ln^2 x+6\ln x-5=0$	[1;3]	Ньютона	1,8832
20	$\sin x^2+\cos x^2-10x=0$	[0;1]	Половинного деления	0,1010
21	$x^2-\ln(1+x)-3=0$	[2;3]	Итераций	2,0267
22	$2x\sin x-\cos x=0$	[0,4;1]	Ньютона	0,6533
23	$e^x+\sqrt{1+e^{2x}}-2=0$	[-1;0]	Половинного деления	-0,2877
24	$\ln x-x+1.8=0$	[2;3]	Итераций	2,8459
25	$\sqrt[3]{x-4}-\frac{1}{x^2+1}=0$	[4;6]	Ньютона	4,0002
26	$e^x-2\cos x=0$	[0;2]	Половинного деления	0,5398
27	$\sqrt[3]{x+2}-3x+16=0$	[4;7]	Итераций	6,0000
28	$\frac{\pi\sin x}{x}-3\cos x-2=0$	[1;2]	Ньютона	1,5708

ОЦЕНОЧНОЕ СРЕДСТВО № 5

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ПРОВЕРОЧНОЙ РАБОТЫ

ВАРИАНТ 1

Составьте программу вычисления значения функции, используя полное ветвление:

$$y=\begin{cases}\frac{4x^2+1}{x-5}, & \text{если } x<5,\\ 3x^2-2, & \text{если } x\geq 5.\end{cases}$$

ВАРИАНТ 2

Составьте программу вычисления значения функции, используя полное ветвление:

$$y = \begin{cases} \dfrac{5x^2+2}{x+4}, & \text{если } x > -4, \\ 3x^2+7, & \text{если } x \le -4 \end{cases}$$

ВАРИАНТ 3

Составьте программу вычисления значения функции, используя полное ветвление:

$$y = \begin{cases} \dfrac{7x^2-1}{2x+6}, & \text{если } x < -3, \\ 4x^2-5, & \text{если } x \ge -3 \end{cases}$$

ВАРИАНТ 4

Составьте программу вычисления значения функции, используя полное ветвление:

$$y = \begin{cases} \dfrac{9x^2+5}{3x+12}, & \text{если } x < -4, \\ 4x^2-7, & \text{если } x \ge -4 \end{cases}$$

ОЦЕНОЧНОЕ СРЕДСТВО № 6

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ПРОВЕРОЧНОЙ РАБОТЫ

Вариант 1

1. Составить программу вычисления суммы и произведения четных чисел из промежутка от 1 до 10.

Вариант 2

2. Составьте программу вывода на экран всех нечетных трехзначных чисел.

Вариант 3

3. Дано целое число N(больше 0). Найти сумму 1+1/2+1/3.....1/N(вещественное число)

ОЦЕНОЧНОЕ СРЕДСТВО № 7

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ОПРОСА

1) Оператор цикла с предусловием, формат команды.
2) Заголовок и тело цикла с предусловием.
3) Оператор цикла с параметром, формат команды.
4) Заголовок и тело цикла с параметром.
5) Шаг изменения счетчика цикла с параметром.
6) Оператор цикла с постусловием, формат команды, условие и тело цикла.
7) Организация повтора программы с использованием цикла с постусловием.
8) Использование составного оператора в теле цикла.

ОЦЕНОЧНОЕ СРЕДСТВО № 8

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ тестирования

Вопрос № 1. Укажите пространство имен, включающее все операции ввода-вывода.

a) System.Files
b) System.IO

c) Filesystem.Main
d) System.Operations

Вопрос № 2. Укажите название метода, который возвращает имена файлов (с указанием пути к ним) в указанном каталоге.
Введите ответ:

Вопрос № 3. Укажите класс, реализующий работу с каталогами в C++.
a) Filesystem
b) Folder
c) Directory
d) Нет верного ответа

Вопрос № 4. Укажите название метода, который возвращает имена подкаталогов (включая их пути) в указанной директории.
Введите ответ:

Вопрос № 5. Какие методы позволяют добавлять данные в текстовый файл?
a) WriteFile
b) WriteLine
c) WriteText
d) Write

Вопрос № 6. Укажите метод, который выполняет чтение строки символов из текущего потока и возвращает данные в виде строки.
a) GetString
b) GetLine
c) ReadString
d) ReadLine

Вопрос № 7. Укажите класс, который реализует чтение текстовых файлов в C++.
a) FileReader
b) StreamReader
c) TextReader
d) DataReader

Вопрос № 8. Укажите класс, который реализует запись текстовых файлов с C++.
a) StreamWriter
b) DataWriter
c) FileWriter
d) TextWriter

Вопрос № 9. Укажите класс, который реализует получение информации о файле в C++.
a) FileInfo
b) DataInfo
c) FileDescription
d) FileProperties

Вопрос № 10. Укажите класс, который отвечает за отслеживание изменений в файловой системе в C++.
a) FileSystemCatch
b) FileSystemWatcher
c) FileSystemInfo
d) FileSystemObserver

Вопрос № 11. Укажите имена файлов, соответствующие формату коротких имен 8.3.
a) FileNa~.exe
b) FileNa~.ex
c) FileNa.exe

d) FileNa*.ex*

Вопрос № 12. Укажите исключение, возникающее при отсутсвии у непосредственно вызывающего класса или производного класса разрешения полного доверия при работе с файловой системой.

Введите ответ:

Вопрос № 13. Какие виды изменений в файловой системе, которые можно отслеживать в C++?

a) Changed
b) Edited
c) Created
d) Written

Вопрос № 14. Какие свойства позволяют получить соответствующую информацию о файле?

a) GetName
b) Length
c) GetLength
d) CreationTime

ОЦЕНОЧНОЕ СРЕДСТВО № 9

1) Описание одномерного массива.
2) Использование операторов цикла при решении задач обработки массивов.
3) Выделение отдельного элемента одномерного массива.
4) Поиск элементов в массиве, кратных заданному числу (условие кратности). Привести фрагмент программы.
5) Описание двумерного массива.
6) Использование операторов цикла при решении задач обработки массивов.
7) Выделение отдельного элемента двумерного массива.
8) Свойства диагональных элементов квадратной матрицы.

ОЦЕНОЧНОЕ СРЕДСТВО № 10

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ОПРОСА

1) Понятие подпрограммы, назначение подпрограмм.
2) Виды подпрограмм в языке программирования.
3) Состав подпрограммы.
4) Вызов подпрограммы.
5) Функции, описание функции, заголовок функции, тело функции.
6) Процедуры, описание процедуры, заголовок процедуры, тело процедуры.
7) Параметры-значение и параметры-переменные процедуры.
8) Формальные и фактические параметры, работа механизма передачи параметров.

ОЦЕНОЧНОЕ СРЕДСТВО № 11

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ОПРОСА

1) Структурированный тип запись, определение, назначение.
2) Описание типа запись, используемые служебные слова.
3) Компоненты записи и их типы.
4) Массив записей, описание, назначение.
5) Составные имена.

6) Оператор присоединения и его область действия.

ОЦЕНОЧНОЕ СРЕДСТВО № 12

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ СОЗДАНИЯ И ЗАЩИТЫ ПРЕЗЕНТАЦИИ

Вопросы, которые необходимо раскрыть при создании презентации:
1) Описание файловых переменных.
2) Установление связи между программой и файлом на диске.
3) Открытие файла для чтения, записи, добавления данных.
4) Запись данных в файл, чтение данных из файла.
5) Завершение работы с файлом.

ОЦЕНОЧНОЕ СРЕДСТВО № 13

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ПРОВЕРОЧНОЙ РАБОТЫ

Вариант 1

1. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все вхождения первой буквы этого слова (количество пробелов между словами не изменять).
2. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Определить количество слов, которые начинаются и заканчиваются одной и той же буквой.
3. В языке используется латинский алфавит. Глагол прошедшего времени получается из глагола настоящего времени изменением порядка следования гласных (a, o, u, i, e) на обратный. Согласные остаются на своих местах. Например, глагол padbote преобразуется в pedbota. Задается глагол настоящего времени. Преобразовать его в глагол прошедшего времени и напечатать.

Вариант 2

1. Дана строка-предложение из символов латинского алфавита. Вывести самое короткое слово в предложении (если таких слов несколько, то вывести первое из них).
2. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Определить количество слов, которые содержат ровно три буквы ' A '.
3. Дана строка из символов латинского алфавита. Проверьте правильность расстановки тега `<td>`: каждому открытому тегу должен соответствовать закрытый `</td>`.

Вариант 3

1. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Определить длину самого длинного слова.
2. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, но разделенные одним символом '.' (точка). В конце точку не ставить.
3. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или несколькими). Преобразовать каждое слово в строке, удалив из него все вхождения последней буквы этого слова (количество пробелов между словами не изменять).

Вариант 4

1. Предложение состоит из слов, разделенных одним или несколькими пробелами. Написать программу, печатающую все слова, оканчивающиеся на заданный символ.
2. В предложении, состоящем из слов, разделенных одним пробелом, заменить первую букву у слов, следующих за словами die, der, das, на прописную.
3. Дана строка, состоящая из символов латинского алфавита, разделенных пробелами (одним или не- сколькими). Преобразовать каждое слово в строке, удалив из него все вхождения заданного символа (количество пробелов между словами не изменять).

ОЦЕНОЧНОЕ СРЕДСТВО № 14

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ подготовки докладов

Тема 3.1. Язык программирования С

Темы докладов:

Язык программирования С. История. Возможности.
Биография Д.Ритчи.

ОЦЕНОЧНОЕ СРЕДСТВО № 15

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ КОМПЬТЕРНОГО ТЕСТИРОВАНИЯ

Тема 3.2. Основные операторы языка программирования С

if (x%2=0)
if (x%2==2)
if (x mod 2 =0)
if (x%2==0)

2.
int mark;
printf("Vvedite ocenku"); scanf("%d",&mark);
if (mark == 4) printf("Vi udarnik.\n");
else if (mark > 4) printf("Vi otlichnik!\n");
else printf ("Ychitsya nado.\n");

С клавиатуры вводится 3, какой результат будет на экране?

Vi otlichnik!
Vi udarnik.
Ychitsya nado.

3. Значение вещественной переменной нужно вывести следующим образом: выравнивание по левому краю, три знака после запятой. Выберите соотвестствующую команду.

printf("%f5.3",x);
printf("%d3",x);
printf("%f-8.3",x);
printf("%-3f",x);

4. Укажите правильно заданную команду для вывода вещественной переменной a.

printf("%f",a);
scanf("%d",&a);
scanf("%f",&a);
printf("%d",a);

5. &&

логическое "И"
логическое "ИЛИ"
логическое "НЕ"

6. ||

логическое "И"
логическое "ИЛИ"
логическое "НЕ"

7. !

логическое "И"
логическое "ИЛИ"
логическое "НЕ"

8. "не равно" в СИ обозначается

==
<>
><
!=

9. Чему равен результат pow(3,3)%5=

4.
5.
2.
3.

10. Описание вещественного типа с СИ

float
int
real
char

11. Какой спецификатор нужно использовать для вывода вещественного числа?

%d
%f
%c
%i

12. Какой спецификатор нужно использовать для вывода целого числа?

%d
%f
%c
%s

13. Какой спецификатор нужно использовать для вывода одного символа?

%d
%f
%c
%i

14. Требуется вывести вещественное число в СИ, так чтобы после запятой было 3 знака. Какой формат нужно использовать?

%3d
%3f
%4.3f
%4.3d

15. В каком формате появится число на экране при использовании команды:
printf("%4.2f",6);

6.
6.00
6.00000
6.000000

16. Какой оператор в СИ находит остаток от деления?

/
mod
%
&

17. Оператор получения адреса

%
&&
&
\n

18. \n в СИ обозначает:

горизонтальная табуляция
перевод курсора на новую строку
спецификатор целочисленного формата
спецификатор вещественного формата

19. Оператор соотношения "равно" записывается:

=
:=
!=
==

20. Как записывается логическое "И" в СИ?

and
&
&&
i

21. Как записывается логическое "ИЛИ" в СИ?

ili
OR
|
||

22. Как записывается логическое "НЕ" в СИ?

|
!
not
&

23. Язык СИ разработал…

Том Куртц
Никлаус Вирт
Деннис Ритчи
Билл Гейтс

24. Чему равен результат выполнения операции 15%3=…

0.
1.
5.
3.

25. Какую библиотеку нужно подключить для работы операторов scanf() и printf() в СИ?

26. Какую библиотеку нужно подключить для работы оператора fabs() в СИ?

27. Как находится модуль вещественного числа в СИ?

mod
fabs
abs
| |

28. Как записать на СИ "корень квадратный из x"?

x ^ 0.5
power(x,2)
pow(x,1/2)
pow(x,y,2)

29. Оператор форматированного вывода:

scanf()
print()
printf()
write()

30. Оператор ввода:

scanf()
read()
printf()
input()

31. Укажите правильно заданную команду для ввода вещественной переменной a.

printf("%f",a);
scanf("%d",&a);
scanf("%f",&a);
scanf("%-f",a);

ОЦЕНОЧНОЕ СРЕДСТВО № 16

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ПРОВЕРОЧНОЙ РАБОТЫ
Тема 3.3. Массивы

Вариант 1
Написать программу на языке программирования C++. В массиве из 20 целочисленных элементов, заданных случайным образом из отрезка [-15;15] найти максимальный элемент и его номер.
Вариант 2
Написать программу на языке программирования C++.В массиве из N(вводится с клавиатуры) целочисленных элементов, заданных случайным образом из отрезка [-5;5] подсчитать количество нечетных положительных элементов, вывести найденные элементы на печать.
Вариант 3
Написать программу на языке программирования C++, которая определяет в целочисленной матрице, состоящей из элементов из отрезка [-5;5], номер строки, которая содержит наибольшее количество элементов, равных нулю.

ОЦЕНОЧНОЕ СРЕДСТВО № 17

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ТЕСТИРОВАНИЯ
(рубежный контроль)

16. Какой оператор в СИ находит остаток от деления?
/
mod
%
«&&»

17. Описание строки s из 10 символов в СИ нужно выполнить:

string s(10);
char s[10];
char s[11];
string s[10];

18. \n в СИ обозначает:

горизонтальная табуляция
перевод курсора на новую строку
спецификатор целочисленного формата
спецификатор вещественного формата

19. Оператор соотношения "равно" записывается:

=
:=
!=
==

20. Функция strlen() …

вычисляет длину строки
выполняет сцепление строк
проверяет вхождение одной строки в другую
выполняет преобразование числа в строку

21. Как записывается логическое "ИЛИ" в СИ?

ili
OR
|
||

22. Как записывается логическое "НЕ" в СИ?
|
!
not
&&
23. Язык СИ разработал…

Том Куртц
Никлаус Вирт
Деннис Ритчи
Билл Гейтс

24. 15%3=…

0.
1.
5.
3.

25. Какую библиотеку нужно подключить для работы операторов scanf() и printf() в СИ?

26
Какую библиотеку нужно подключить для работы оператора fabs() в СИ?

27. Как находится модуль вещественного числа в СИ?

mod
fabs
abc
| |

28. Как записать на СИ "корень квадратный из x"?

x ^ 0.5
power(x,2)
pow(x,0.5)
pow(x,y,2)

29. Оператор форматированного вывода:

scanf()
print()
printf()
write()

30. Оператор ввода:

scanf()
read()
printf()
input()

31. Укажите правильно заданную команду для ввода вещественной переменной a.

printf("%f",a);
scanf("%d",&a);
scanf("%f",&a);
scanf("%-f",a);

32. Укажите правильно заданную команду для вывода вещественной переменной a.

printf("%f",a);
scanf("%d",&a);
scanf("%f",&a);
printf("%d",a);

33. Значение вещественной переменной нужно вывести следующим образом: выравнивание по левому краю, три знака после запятой. Выберите соответствующую команду.

printf("%f5.3",x);
printf("%d3",x);

printf("%f-8.3",x);
printf("%-3f",x);

34. Служебное слово switch используется в конструкции…

условия
множественного выбора
цикла с параметром
форматированного вывода

35. Выберите правильно записанное условие проверки числа x на четность на языке СИ:

if (x%2=0)
if (x%2==2)
if (x mod 2 =0)
if (x%2==0)

36. Переход на новую строку в языке Си осуществляется:

\t
%d
\n
&&

37. Укажите количество ошибок при использовании условного оператора на языке Си:

```
if (a==5)
{
printf(“Good”);
b=a;
}
```

1.
2.
3.
нет ошибок

38. Укажите неправильный вариант для обозначении имен переменных:
A: VAZ
B: _sww
C: 5a
D: re4
E: kl*s

C и E
Только E
Все неправильные
Все правильные

39. Каким словом обозначается «главная» функция в программе на языке Си:

mail
main
#include
return

40. Укажите какое значение будет выведено на экран:
h=4.3767;
j=0;
printf(“%1.1f”,h);

4.0
0.3767
0.0
4.4

41. Укажите значение элемента массива K[2] после завершения цикла:
int K[78];
for(i=0;i<=34;i+=2)
K[i]=i-10;

-8
2
34
-10

42. Укажите количество ошибок в строке, подключающей модуль для работы операторов ввода и вывода: #incude
1.
2.
3.
нет ошибок.

43. Целый тип в языке Си обозначается:

integer
float
int
Такого типа не существет

44. Укажите количество ошибок при использовании условного оператора на языке Си:
if (a<-3);
{
printf(“%d\n”,a);
a++;
}

1.
2.
3.
нет ошибок.

45. Укажите количество ошибок при использовании оператора цикла на языке Си:
```
for(t=-4,t<4;t=t+0.5
{
printf(“%f\n”,t);
}
```

1.
2.
3.
4.

46. Укажите неправильный вариант для обозначении имен переменных:
A: a45
B: y_22
C: рус
D: f%
E: _1

B
C
D
C и D

47. Подключение какой библиотеки необходимо для использования математических функций:

math.h
stdio.h
#include
conio.h

48. Укажите какое значение будет выведено на экран:
```
h=-6.278;
j=6;
printf(“%4.2f”,j);
```

6.00
-6.28
-6.00
6.28

49. Укажите значение элемента массива K[13] после завершения цикла:
```
int K[13];
```

```
for(i=0;i<13;i=i++)
K[i]=i;
```

13.
0.
Массив задан неправильно
Элемент K[13] не определится

50. Укажите количество ошибок в строке:
```
print(“%d %d”,y,&y);
```

1.
2.
3.
Ошибок нет.

51. Для подключения модулей используется:

#include
stdio.h
main
getch()

52. Тип переменной указывается:
через : после переменной
после переменной
указывать не обязательно
до ее написания

53. Укажите количество ошибок при использовании условного оператора на языке Си:
```
if (c<>3)
{
G=sin(c);
printf(“%f\n”,G);
c--;
};
```

1.
2.
3.
Ошибок нет.

54. Укажите количество ошибок при использовании оператора цикла на языке Си:
```
for (g=0,g<13,g++);
{
printf(“%f\n”,g);
}
```
1.

2.
3.
4.

55. Подключение какой библиотеки необходимо для использования функции getch()?
stdio.h
stdlib.h
getch.h
conio.h

56. Укажите значение элемента массива K[0] после завершения цикла:
```
int K[4];
for(i=3;i>0;i=i-1)
K[i]=i*i;
K[0]=K[2];
```

4.
2.
0.
1.

57. Укажите количество ошибок при использовании условного оператора:
```
if e=12
printf("%d",e);
```

1.
2.
4.
Нет ошибок.

58. Для использования функции strlen() необходимо подключить библиотеку:

stdio.h
conio.h
string.h
stdlib.h

59. Укажите диапазон случайных чисел для random(12)-6:

от 0 до 6
от -6 до 0
от -6 до 5
от -6 до -1

60. Укажите количество ошибок при использовании условного оператора на языке Си:
```
while (d!=13)
{
G=cos(sqrt(d));
printf(“%f\n”,G);
d--;
}
```

1.
2.
3.
Нет условного оператора.

61.
```
while (d!=13)
{
G=cos(sqrt(d));
printf(“%f\n”,G);
d--;
}
```
Какой оператор используется в коде программы?

оператор условия
оператор цикла с параметром
оператор цикла с предусловием
оператор цикла с постусловием

62. Укажите значение элемента массива K[4] после завершения цикла:
```
int K[100];
for(i=3;i<100;i++)
{
if(i==4)
K[i]=10;
else K[i]=2*i;
}
```

100.
4.
8.
10.

63. Каким символом завершается программа на языке Си:
«}»
«getch»
«.»
«return»

64. Можно ли в операторе условия if не ставить скобки {}

Всегда надо ставить
Можно, если нет else
Нельзя
Можно, если блок состоит только из одного оператора

65. Целочисленный массив обозначается:

float A[4]
int A[13]
int A[1][3]
2 или 3 вариант

66. Укажите количество ошибок при использовании условного оператора на языке Си:

```
if (x<10)
{
printf(“%f\n”,x);
x++;
}
else printf(“end”);
```

1.
2.
3.
Нет ошибок.

67. Какой модуль необходимо подключить для функции random?

stdio.h
stdlib.h
conio.h
randomize.h

68. Какой из операторов не является оператором цикла?
for
while
if
1, 2 и 3 варианты

69. Двумерный массив обозначается:

W[3][5]
W[3;5]
W[3,5]
W[3],[5]

70. В каком порядке следует подключать модули?
В порядке следования необходимых функций
В строго определённом
Не имеет значения
Их можно вообще не подключать

71. a=5
b=7
После выполнения операции a+=b значение переменной a равно:

12.1
5.35.
операция не имеет смысла

72.
```
int a,b;
float c;
printf("Vvedite a\n"); scanf("%d",&a);
printf("Vvedite b\n"); scanf("%d",&b);
c=a/b;
printf("c=%3.2f\n",c);
```

С клавиатуры вводятся 12 и 5. Какой результат выведется на экран?

c=0.05
c=2.001
c=2.40
c=3.20

73. В каком выражении использован оператор инкремента?

4*a--
a*=b
x*y++1
ceil(x)

74. Оператор, уменьшающий значение перемннной на единицу - это ...
fabs
инкремента
floor
дикремента

ОЦЕНОЧНОЕ СРЕДСТВО № 19

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ практического задания

2. Вы можете воспользоваться конспектами, учебником
3. Максимальное время выполнения задания: *2 часа.*
4 Перечень раздаточных и дополнительных материалов *методические указания к лабораторным работам*

Оборудование и оснащение: *ПК*

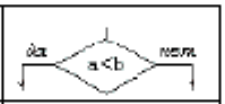
Практическое задание

Цель работы: Научиться составлять циклические алгоритмы, создавать программы, используя полученные знания.

Содержание работы.

Основные понятия.

1 Графический способ представления циклических алгоритмов (метод блок-схемы).

Проверка условий	
Цикл с параметром	

2 Алгоритм, в котором вычисления повторяются по одной и той же совокупности формул, называется циклическим

3 Существуют следующие конструкции для организации циклов:

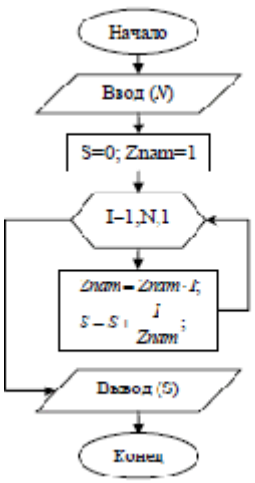
- цикл с предусловием;
- цикл с постусловием;
- цикл со счетчиком.

4 Графическое представление циклических алгоритмов

5 Циклы с предусловием используются тогда, когда выполнение цикла связано с некоторым логическим условием. Оператор цикла с предусловием имеет две части: условие выполнения цикла и тело цикла. При выполнении оператора цикла определенная группа операторов выполняется до тех пор, пока

определенное в операторе условие истинно. Если условие сразу ложно, то оператор не выполнится ни разу.

Задание



1. Начало;
2. Ввод (*N*);
3. S=0;
4. Znam=1;
5. Для I = 1 до N выполнить

нц

$Znam = Znam \cdot I;$

$S = S + \frac{I}{Znam};$

кц

6 Вывод (S);

7. Конец.

Дополнительно на оценку

Задания к практической работе.

1 Даны действительные числа x, y. Вывести в порядке возрастания все целые числа, расположенные между x и y, а также количество этих чисел.

2 Даны действительные числа x, y. Вывести в порядке убывания все целые числа, расположенные между x и y, а также количество этих чисел.

3 Дано действительное число – цена 1 кг конфет. Вывести стоимость 1, 2 … 15 кг конфет.

Составить и записать алгоритм решения задачи в графическом и словесно-формульном виде

Исходные данные:

Дано целое число N. Вычислить значение выражения $1+\frac{2}{2!}+\frac{3}{3!}+...+\frac{N}{N!}$, результат вывести как действительное число.

Решение:

1 $N! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot N$

Значение выражения будем рассчитывать поэтапно.

На первом шаге к значению суммы S добавим $1=\frac{1}{1!}=\frac{1}{1}$

На втором $\frac{2}{2!}=\frac{2}{1 \cdot 2}$

На третьем $\frac{3}{3!}=\frac{3}{1 \cdot 2 \cdot 3}$

…

На N-ном $\frac{N}{N!}=\frac{N}{1 \cdot 2 \cdot \ldots \cdot N}$

В зависимости от заданного пользователем значения N количество шагов в алгоритме может быть разным, поэтому для определения шага добавим промежуточную переменную I, которая будет меняться от 1 до N

Так как к значению суммы на каждом шаге мы добавляем какое-то число, необходимо определить начальное значение, не влияющее на результат. Ясно, что это S = 0.

На каждом шаге считать факториал трудно, поэтому обратим внимание, что на любом этапе вычислений значение факториала отличается от предыдущего на число, равное значению шага, значит введем переменную для факториала: Znam = 1.

Получили общие формулы:

$Znam = Znam \cdot I;$

$S = S + \frac{I}{Znam};$

ОЦЕНОЧНОЕ СРЕДСТВО № 20

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ практического задания

Практическое задание
Массив - это структура данных, что представляет собой совокупность фиксированного размера и конфигурации упорядоченных однородных независимых переменных.

Массив относится к так называемым структурированным данным, то есть таких, что имеют фиксированную внутреннюю структуру (организацию).

Массив характеризуется:

1. Количеством размерностей (количеством координат, необходимых для определения местонахождения нужного элемента массива).

2. Общим идентификатором (именем) для всех элементов массива.

3. Индексом или совокупностью индексов, которые определяют каждый отдельный элемент массива.

Одномерный массив (вектор) - имеет одну размерность

При обращении к отдельному элементу массива необходимо указать его индекс (местонахождение в массиве):

A[7] i:=7; A[i]

Здесь i - индекс элемента массива, может быть только целого или натурального типа Двумерные массивы (матричные) - имеют две размерности, m*n. Доступ к отдельному элементу массива осуществляется путем определения двух его координат: номера строки i=1..m и столбца j=1..n

Операции предоставления выполняются аналогично:

a[3,9]=8; B:=A[1,1]

В трехмерном массиве для доступа к элементу необходимо указать три индекса A[i,j,k]. Можно создавать массивы с большей размерностью, но работа с массивами, размерность которых превышает 3, существенным образом усложняет алгоритм, поэтому, по возможности, необходимо избегать организации подобных структур данных.

Сортировка массивов

Методы сортировки можно разбить в соответствии с определяющими их принципами на три основные группы:

1. Сортировка с помощью вставки (by Insertion) или с помощью включения

2. Сортировка с помощью выбора (by Selection) или с помощью выделения

3. Сортировка с помощью обмена (by Exchange) или пузырьковая.

Каждая группа имеет прямой метод (самый простой) и улучшенный(усложненный) методы сортировки

I. Сортировка с помощью вставки

Принцип сортировки: массив распределяется на отсортированную и неотсортированную части. На первом шаге за отсортированную часть (последовательность) принимается первый элемент массива. Каждый следующий элемент из неотсортированной части вставляем в заранее отсортированную последовательность так, чтобы эта последовательность оставалась отсортированной.

При этом надо:

1. Найти место, куда нужно вставить этот элемент

2. Сдвинуть элементы, которые стоят справа в отсортированной части на одну позицию вправо.

3. На освобожденное место поставить элемент, который анализируется (вставляется).

Два способа выполнения этих действий:

1) каждый следующий элемент сравнивается с элементами в отсортированной части, находится место вставки, все следующие элементы сдвигаются на одну позицию вправо и после этого вставляется элемент;

2) элемент, который вставляется, последовательно, слева направо, сравнивается с любым из элементов в отсортированной части. Если нужно, элемент в отсортированной части сразу сдвигается на одну позицию вправо. Как только найдено нужное место вставки, элемент, который анализируется, вставляется на нужную позицию.

II. Сортировка с помощью прямого выбора

Принцип сортировки: массив также делится на отсортированную и неотсортированную части. На первом шаге весь массив - неотсортированный. В неотсортированной части находится минимальный (или максимальный) элемент и меняется местами с первым эле-

ментом неотсортированной части. Граница отсортированной части сдвигается вправо на 1. Процедура выполняется циклически, n-1 раз (последний элемент передвигать не надо)

III. Сортировка с помощью прямого обмена (пузырьковая)

Принцип сортировки: слева направо поочередно направляется сравнение двух соседних элементов. Если они не упорядочены между собою, то меняются местами. В базовом алгоритме прохождения массива и очередное приведение в порядок повторяются n-1 раз.

Задание №1
Найдите среднее арифметическое первых двухсот чисел.

Дополнительные задания на оценку
Задание № 2
Найдите минимальное число среди введенных 10 чисел.
Задание № 3
Найдите максимальное число среди введенных 15 чисел.

ОЦЕНОЧНОЕ СРЕДСТВО № 1

ПЕРЕЧЕНЬ ТЕОРЕТИЧЕСКИХ ВОПРОСОВ
К ЭКЗАМЕНУ

1. Алгоритм. Базовые алгоритмические конструкции.
2. Исполнители алгоритма. СКИ. Пошаговое выполнение алгоритма.
3. История и классификация языков программирования.
4. Происхождение и достоинства языка C/C++.
5. Структура программы на C/C++. Пример простой программы.
6. Элементы языка C/C++ (алфавит, лексемы языка, идентификатор, ключевые слова, константы, разделители, выражения).
7. Встроенные типы данных (целый тип, числа с плавающей точкой, символьный тип, тип bool).
8. Определение переменных. Операции языка C++ (операция присвоения полная и короткая форма, операции сложения, вычитания, изменения знака, умножения, деления, деление по модулю, операции увеличения и уменьшения).
9. Ввод и вывод данных (функции printf() и scanf()).
10. Условные операции (меньше (больше), меньше (больше) или равно, равенство, неравенство, логическое И и ИЛИ).
11. Структура следования, структуры выбора (if, if/else, switch/case).
12. Структуры повторения (циклы с предусловием while).
13. Структуры повторения (for).
14. Структуры повторения (цикл с постусловием do/while).
15. Операторы перехода (break, continue, return, goto).
16. Локальные и глобальные переменные.
17. Объявление, определение и вызов функции.
18. Передача параметров функции по значению. Возвращаемое значение функции. Прототип функции.
19. Функция с переменным числом параметров.
20. Рекурсия.
21. Классы памяти.
22. Генерация случайных чисел.
23. Понятие указателя. Операции над указателями.
24. Инициализация указателя. Указатель на тип void.
25. Модификатор const. Передача параметров через указатель.
26. Массивы. Одномерные и многомерные массивы.

27. Массивы. Объявление, инициализация массивов, обращение к элементам массива. Определение массива.
28. Связь между указателями и массивами. Операции над указателями.
29. Передача массива в функцию.
30. Строки. Объявление и инициализация массива символов (строк). Нулевой символ. Библиотечные функции работы со строками.
31. Динамическое распределение памяти. Библиотечные функции для выделения и освобождения динамической памяти.
32. Оператор sizeof(). Функции выделения памяти malloc(), calloc().
33. Функция переопределение динамически выделенной памяти realloc(). Функция освобождения памяти free().
34. Переименование типов. Перечисления.
35. Объявление шаблонов структур и объединений. Определение и инициализация структур-переменных.
36. Присвоение структур-переменных. Доступ к полям структуры.
37. Указатели на структуры.
38. Типы файлов: текстовые и бинарные. Внешние файлы.
39. Связывание файловых переменных с внешней средой.
40. Типовые действия с файлами: создание, открытие, закрытие, чтение и изменение.
41. Организация ввода-вывода информации в файл. Последовательный и произвольный доступ к файлу
42. Стандартные функции по работе с файлами.
43. Перегрузка функций. Шаблон функции. Значение формальных параметров по умолчанию в языке C++.
44. Препроцессоры #define, #undef. Макроопределения препроцессора (с параметрами и без). Условная компиляция #if, #ifdef, #ifndef, #else, #endif.

ОЦЕНОЧНОЕ СРЕДСТВО № 2

ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ВОПРОСОВ
К ЭКЗАМЕНУ

1) Составить программу для вычисления значения выражения:

$$y = \cos^2 2x + \left|\sin \frac{x}{2}\right|$$

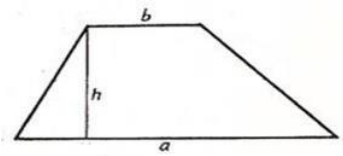
Входные данные – x (вводится с клавиатуры). Выходные данные – y.
2) Составьте блок-схему алгоритма и программу, определяющую является ли введенное число двузначным. Результатом работы программы должно быть сообщение 'является' или 'не является'.
3) Составьте блок-схему и программу вычисления значения функции, используя алгоритм полного ветвления:

4) Составьте блок-схему алгоритма и программу вычисления площади и периметра прямоугольника, если известны его стороны (вводятся с клавиатуры).
5) Составьте блок-схему алгоритма и программу вычисления площади и периметра треугольника, если известны его стороны и высота (вводятся с клавиатуры).
6) Составьте блок-схему алгоритма и программу, определяющую является ли введенное целое число положительным. Результатом работы программы должно быть сообщение 'является' или 'не является'.

7) Составьте блок-схему алгоритма и программу, определяющую является ли введенное целое число нечетным. Результатом работы программы должно быть сообщение 'является' или 'не является'.
8) Составьте блок-схему алгоритма и программу вычисления количества букв в строке. Строка вводится с клавиатуры.
9) Составьте блок-схему алгоритма и программу вычисления площади ромба по его диагоналям (вводятся с клавиатуры).

$$S=\frac{1}{2}d_1 \cdot d_2$$

10) Составьте блок-схему алгоритма и программу вычисления площади трапеции по ее основаниям и высоте (вводятся с клавиатуры).



$$S=\frac{a+b}{2}h$$

11) Составить блок-схему алгоритма и программу для вычисления значения выражения: Входные данные – c, b (вводятся с клавиатуры). Выходные данные – a.

$$а)\ y=\frac{5x^2-4}{7x+2};\qquad б)\ a=\sqrt{\frac{b+c}{5bc}}-\frac{b^2}{2c};$$

$$в)\ y=\sin^2\frac{x}{2}-\left|\cos 2x\right|;\qquad г)\ x^2+y^2\geq(z+2)^2;$$

$$д)*\ k_1=\sqrt{\frac{8x^2-3}{\left|2x^2y\right|}}-\frac{(x+y)^2}{2xy}.$$

12) Составьте блок-схему алгоритма и программу, которая по введенному с клавиатуры целому числу в диапазоне 0 – 9 выводит строку — название соответствующей цифры на русском языке (0 — "ноль", 1 — "один", 2 — "два", ...).
13) Составьте блок-схему алгоритма и программу вычисления среднего арифметического трех целых чисел, введенных с клавиатуры, с точностью до трех знаков после запятой.
14) Составить блок-схему алгоритма и программу для вычисления значения выражения: Входные данные – x,y (вводятся с клавиатуры). Выходные данные – s.

$$s=\left|\frac{x+\sqrt{y}}{x^2+y^2}\right|$$

15) Составить блок-схему алгоритма и программу, которая по введенному номеру дня недели (вводится с клавиатуры), выдает его название.
16) Составьте блок-схему алгоритма и программу вычисления площади круга и длины окружности по введенному радиусу (вводится с клавиатуры).
17) Составьте блок-схему алгоритма и программу вычисления площади и периметра квадрата, если известны его стороны (вводятся с клавиатуры).
18) Составьте блок-схему алгоритма и программу вывода на экран всех чисел, которые кратны числу 5, в интервале от a до b (значения a и b вводятся, a

19) Составьте блок-схему алгоритма и программу вычисления произведения целых чисел из промежутка [-6; 5).
20) Составьте блок-схему алгоритма и программу вывода на экран всех трехзначных чисел, кратных трем. Решите данную задачу, используя цикл с параметром, цикл с предусловием и цикл с постусловием.
21) Составьте блок-схему алгоритма и программу расположения трех чисел в порядке возрастания.
22) Составьте блок-схему алгоритма и программу вывода на экран таблицы степеней 2^n, где $0 \le n \le 10$.
23) Составьте блок-схему алгоритма и программу вывода таблицы значений функции $y = 4x^2 + 5x - 10$ на отрезке [-9; 9] с шагом h = 3. Вывод результатов оформите в виде таблицы с границами.
24) Составьте блок-схему алгоритма и программу вычисления среднего арифметического целых чисел из отрезка [-4; 15] с точностью до трех знаков после запятой.
25) Составьте блок-схему алгоритма и программу вывода таблицы значений функции $y = x^2 - 5x - 2$ на отрезке [1; 20] с шагом h = 2. Вывод результатов оформите в виде таблицы с границами.
26) Составьте блок-схему алгоритма и программу вычисления значения выражения для данного натурального числа N: $1/1^2 + 1/2^2 + 1/3^2 + \ldots + 1/N^2$. Значение N вводится с клавиатуры.
27) Составьте блок-схему алгоритма и программу заполнения массива, не используя клавиатуру, числами: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20. Найти сумму элементов полученного массива.
28) Составьте блок-схему алгоритма и программу, определяющую, на какую букву начинается второе слово в строке, введенной с клавиатуры.
29) Составить программу и блок-схему алгоритма. Сгенерировать массив A из 10 целых чисел, которые берутся из промежутка [-15,15]. Подсчитать количество нечетных элементов массива.
30) Составить программу и блок-схему алгоритма. Сгенерировать массив C из 15 целых чисел, которые берутся из промежутка [-10,10]. Подсчитать сумму четных элементов массива.

IV. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНИИВАНИЯ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ, ХАРАКТЕРИЗУЮЩИЕ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

Процедура оценивания – порядок действий при подготовке и проведении аттестационных испытаний и формировании оценки.

Процедура промежуточной аттестации проходит в соответствии с Положением о промежуточной (рубежной) аттестации знаний студентов и учащихся ФПОУ МОКСУ.

Аттестационные испытания проводятся преподавателем (или комиссией преподавателей – в случае модульной дисциплины), ведущим лекционные занятия по данной дисциплине, или преподавателями, ведущими практические и лабораторные занятия (кроме устного экзамена). Присутствие посторонних лиц в ходе проведения аттестационных испытаний без разрешения ректора или проректора не допускается (за исключением работников университета, выполняющих контролирующие функции в соответствии со своими должностными обязанностями). В случае отсутствия ведущего преподавателя аттестационные испытания проводятся преподавателем, назначенным письменным распоряжением по кафедре.

Инвалиды и лица с ограниченными возможностями здоровья, имеющие нарушения опорно-двигательного аппарата, допускаются на аттестационные испытания в сопровождении ассистентов-сопровождающих.

Во время аттестационных испытаний обучающиеся могут пользоваться программой учебной дисциплины, а также с разрешения преподавателя справочной и нормативной литературой, непрограммируемыми калькуляторами.

Время подготовки ответа при сдаче зачета/экзамена в устной форме должно составлять не менее 40 минут (по желанию обучающегося ответ может быть досрочным). Время ответа – не более 15 минут.

При подготовке к устному экзамену экзаменуемый, как правило, ведет записи в листе устного ответа, который затем (по окончании экзамена) сдается экзаменатору.

При проведении устного экзамена экзаменационный билет выбирает сам экзаменуемый в случайном порядке.

Экзаменатору предоставляется право задавать обучающимся дополнительные вопросы в рамках программы дисциплины текущего семестра, а также, помимо теоретических вопросов, давать задачи, которые изучались на практических занятиях.

Оценка результатов устного аттестационного испытания объявляется обучающимся в день его проведения. При проведении письменных аттестационных испытаний или компьютерного тестирования – в день их проведения или не позднее следующего рабочего дня после их проведения.

Результаты выполнения аттестационных испытаний, проводимых в письменной форме, форме итоговой контрольной работы или компьютерного тестирования, должны быть объявлены обучающимся и выставлены в зачётные книжки не позднее следующего рабочего дня после их проведения.

Порядок подготовки и проведения промежуточной аттестации в форме зачета/экзамена

действие	сроки	методика	ответственный
выдача вопросов для промежуточной аттестации	1 неделя семестра	на лекционных /практических и др.занятиях, на офиц.сайте вуза и др.	ведущий преподаватель
консультации	последняя неделя семестра/период сессии	на групповой консультации	ведущий преподаватель
промежуточная аттестация	в период сессии	устно, письменно, тестирование бланочное или компьютерное, по билетам, с практическими заданиями	ведущий преподаватель, комиссия

формирование оценки	на аттестации		ведущий преподаватель, комиссия