МИНИСТЕРСТВО ОБРАЗОВАНИЯ СТАВРОПОЛЬСКОГО КРАЯ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
"НЕФТЕКУМСКИЙ РЕГИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ"

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ПРОВЕДЕНИЯ САМОСТОЯТЕЛЬНОЙ ВНЕАУДИТОРНОЙ РАБОТЫ ДЛЯ СТУДЕНТОВ

по МДК 01.01 Системное программирование

ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем

Для специальности: 09.02.03 ПРОГРАММИРОВАНИЕ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

2019 г.

ОДОБРЕНО:
НА ЗАСЕДАНИИ ПМО
специальностей
09.02.03«Программирование в компьютерных системах»,
09.02.02«Компьютерные сети» и профессии 09.01.03 «Мастер по обработке цифровой информации»
ПРОТОКОЛ №_1_
«28» августа2019 г.
Руководитель ПМО
_____/И.А.Мазяр /

Методические указания составлены в соответствии с требованиями Федерального государственного образовательного стандарта среднего профессионального образования по специальности **09.02.03 Программирование в компьютерных системах**
УТВЕРЖДАЮ:
Заместитель директора по учебно-методической работе
_____/Е.С.Шведова /
ПОДПИСЬ (ФИО)

Составитель: Усенко Анна Геннадьевна, преподаватель ГБПОУ НРПК

Рецензенты: Мазяр Ирина Анатольевна, преподаватель ГБПОУ НРПК

01.01 Системное программирование разработаны в соответствии с программой специальности среднего профессионального образования (СПО)
09.02.03 Программирование в компьютерных системах

Разработчик:
Усенко А.Г– преподаватель ГБПОУ НРПК

Рассмотрены и одобрены профессионально методическим объединением педагогов специальностей 09.02.03 «Программирование в компьютерных системах», 09.02.02 «Компьютерные сети» и профессии 09.01.03«Мастер по обработке цифровой информации»

Протокол заседания №1 от «28» августа 2019 г.

Пояснительная записка

МДК.01.01 Системное программирование входит в состав ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем. Самостоятельная работа является одним из видов внеаудиторной учебной работы обучающихся.

Основные цели самостоятельной работы:

- систематизация и закрепление теоретических знаний и практических умений обучающихся;
- углубление и расширение теоретических знаний;
- формирование умений использовать справочную документацию и дополнительную литературу;
- развитие познавательных способностей и активности обучающихся, творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельного мышления;
- развитие исследовательских умений.

Особую важность приобретают умения обучающихся: подбирать материалы для профессиональной деятельности, разбираться в текстах на языке программирования Ассемблер, писать небольшие программные модули.

Рекомендации для обучающихся по выработке навыков самостоятельной работы:

Слушать, записывать и запоминать лекцию.

Внимательно читать задание.

Выбрать свой уровень подготовки задания.

Обращать внимание на рекомендуемую литературу.

Из перечня литературы выбирать ту, которая наиболее полно раскрывает вопрос задания.

Обращать внимание на достижение основной цели работы.

Распределение часов на выполнение самостоятельной работы студентов по разделам и темам профессионального модуля ПМ 01; МДК 01.01 «Системное программирование»

Раздел самостоятельной внеурочной работы	Задание самостоятельной внеурочной работы	Трудоемкость	Освоение ОК, ПК
Раздел 1 Основы программирования на алгоритмическом языке	Работа с двоично-десятичными, шестнадцатеричными числами и символами кода ASCII	2	ОК 1-9 ПК 1.2-1.4
	Представление команд процессора. Форматы команд.	2	ОК 1-9 ПК 1.2-1.4
	Работа в отладчике DEBUG: ввод данных разного типа: числовые, символьные	2	ОК 1-9 ПК 1.2-1.4
	Основные понятия языка Ассемблер	2	ОК 1-9 ПК 1.2-1.4
	Ввод, ассемблирование, компоновка, выполнение программы	2	ОК 1-9 ПК 1.2-1.4
	Требования к программе. Ассемблирование, компоновка, выполнение программ	2	ОК 1-9 ПК 1.2-1.4
	Ввод, ассемблирование, компоновка, выполнение программы	2	ОК 1-9 ПК 1.2-1.4
	Ввод, ассемблирование, компоновка, выполнение программ	2	ОК 1-9 ПК 1.2-1.4
	Основные команды языка процессора	2	ОК 1-9 ПК 1.2-1.4

	Ввод, ассемблирование, компоновка, выполнение программ	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Режимы адресации. Определение данных	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
Раздел 2 Основы программирования на машинно-ориентированном языке Ассемблер	Создание EXE-программы. Работа в отладчике AfdPro	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Создание программ с разветвлением	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Создание программ с использованием циклов	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Создание программ с использованием циклов	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Понятие процедуры. Команды логических операций	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Создание программ с использованием логических операций	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Команды сдвигов	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Создание программ с использованием команд сдвигов	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
	Составление программ с использование ввода-вывода на экран.	Письменная самостоятельная работа (в электронном виде)	2	ОК 1-9 ПК 1.2-1.4
Всего				40

Раздел самостоятельной внеурочной работы	Задание самостоятельной внеурочной работы	Трудоемкость	Освоение ОК, ПК
Раздел 3 Основы программирования на процедурном алгоритмическом языке СИ	1.Изучить дополнительную литературу и составить информационное сообщение на тему: «Язык программирования высокого уровня Си». «Язык программирования С++» «Популярные языки программирования для создания операционных систем»	2	ОК 1-9 ПК 1.2-1.4
	2.Составить сравнительную таблицу достоинств и недостатков языков программирования	2	ОК 1-9 ПК 1.2-1.4
	1. Создать презентацию по темам: «Алгоритмизация и требования к алгоритму» «Основные этапы разработки программного приложения» «Структурное программирование» «Отладка и тестирование программы» «Программное приложение С++ Builder»	2	ОК 1-9 ПК 1.2-1.4
	2. Создать программное приложение арифметического выражения для заданных значений *x*, *y*, *z*, по вариантам.	6	ОК 1-9 ПК 1.2-1.4
	1. Написать и отладить программу вывода всех значений функции *S*(*x*) для аргумента *x*, изменяющегося в интервале от *a* до *b* с шагом *h* и заданном *n*, по вариантам.	6	ОК 1-9 ПК 1.2-1.4
	1. Написать и отладить программу обработки одномерных массивов, по вариантам.	6	ОК 1-9 ПК 1.2-1.4
	1.Написать и отладить программу обработки двухмерных динамических массивов, по вариантам.	6	ОК 1-9 ПК 1.2-1.4
	1. Изучить дополнительную литературу и подготовить информационное сообщение на	2	ОК 1-9 ПК 1.2-1.4

	темы: «Распределение памяти» «Аппаратные прерывания» «Управление видеоадаптером» **«**Распределение памяти, логическая структура диска и физический дисковый адрес»		
	1. Оформить практические задания в отчет.	2	ОК 1-9 ПК 1.2-1.4
	1. Написать и отладить программу «Оконного приложения».	6	ОК 1-9 ПК 1.2-1.4
	1. Составить и отладить программу «Обработка списка записей».	6	ОК 1-9 ПК 1.2-1.4
	1. Изучить дополнительную литературу и подготовить информационное сообщение на темы: «Распределение памяти DOS» «Управление программами» «Дисковая структура DOS»	6	ОК 1-9 ПК 1.2-1.4
	1. Оформить практические задания в отчет и подготовится к защите.	6	ОК 1-9 ПК 1.2-1.4
	Всего	58	

Раздел 1 Основы программирования на алгоритмическом языке

Самостоятельная работа №1

Название работы: решение примеров на действия с двоично-шестнадцатиричными числами.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа в тетради.
Количество часов на выполнение: 2

Задание:

1. Перевести числа из десятичной системы в двоичную:

   12, 45, 123, 372, 512, 1024

2. Перевести числа из десятичной системы в шестнадцатеричную:

   15, 23, 85, 345, 1084

3. Перевести числа из двоичной системы в десятичную:

   0001 B, 0001 0111 B, 0010 0101 B

Критерий оценки:
правильно выполнены 3 задания – оценка «5»правильно выполнены 2 задания – оценка «4»правильно выполнено 1 задание – оценка «3»

Самостоятельная работа №2

Название работы: представление команд процессора. Форматы команд.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа в тетради.
Количество часов на выполнение: 2

Задание:

1. Перевести числа из двоичной системы в шестнадцатеричную:

   0011 B, 0111 0010 B, 0111 0000 B

2. Перевести числа из шестнадцатеричной системы в двоичную:

   FB, 01 C4, A5 E7

3. Перевести числа из шестнадцатеричной системы в десятичную:

   C6, A3 FD, 01 C2

Критерий оценки:
правильно выполнены 3 задания – оценка «5»правильно выполнены 2 задания – оценка «4»правильно выполнено 1 задание – оценка «3»

Самостоятельная работа №3, 4

Название работы: работа в отладчике DEBUG: ввод данных разного типа:числовые, символьные.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).
Количество часов на выполнение: 4
Задание:

1. Как будут выглядеть в памяти машины IBM PC числа и символы – 328,1110011101101001b, 95 , @, {, если они расположатся там, начиная с адреса FFEC. Представить числа в виде 2-х байтов.

2.а) Составить программу в машинных кодах:

- занести в регистр AX десятичное число -184;

- прибавить десятичное число 15 к AX;
- переслать содержимое AX в BX;
- прибавить AX к BX;
- почистить AX;
- выход в DOS.

б) Записать программу в машинных кодах в память со смещением 100в) Рассмотреть содержимое всех регистров.

г) Рассмотреть записанную программу в памяти.

д) Осуществить пошаговое выполнение созданной программы до команды RET.

3.

а) Составить программу в машинных кодах:

- переслать слово (число 34) - два байта, начинающиеся в сегментеданных с адреса 04 в регистр AX;
- прибавить содержимое слова (второе число 12) - два байта, начинающиеся в сегменте данных с адреса 02 к регистру AX;
- переслать содержимое регистра AX в слово, начинающиеся в сегменте данных DS с адреса 00;
- вернуться в DOS.

б) Рассмотреть содержимое сегмента данных.

в) Рассмотреть записанную программу в памяти.г) Рассмотреть содержимое всех регистров.

д) Осуществить пошаговое выполнение созданной программы до команды RET.

4. Как будут выглядеть в памяти машины IBM PC числа и символы - 234,0111011b, 176 , &, #, если они расположатся там, начиная с адреса 1EFA. Представить числа в виде 2-х байтов

Критерий оценки:

правильно выполнены 4 задания – оценка «5» правильно выполнены 3 задания – оценка «4»правильно выполнены 2 задания – оценка «3»

Раздел 2Основы программирования на машинно-ориентированном языке Ассемблер

Самостоятельная работа №5-11

Название работы: создание простых программ на языке Ассемблер.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).

Количество часов на выполнение: 13

Задание 1. Написать программу на языке программирования Ассемблер, которая заносит число 5 в регистры AX, BX, CX, DX. Создать объектный,выполняемый файл просмотреть EXE файл в отладчике.

Задание 2.Составить программу на языке АССЕМБЛЕРА, задавая все определения сучетом того, что выполняемый модуль должен иметь расширение .EXE.

- в сегменте данных задать следующие числа и символьные выражения:Фамилия, через запятую - Имя, Отчество, возраст, номер дома, номер квартиры,

любое двоичное число>16,

любое шестнадцатеричное число в интервале от 10 до 1000,по возможности задать четыре последних числа в одном байте,

в двух байтах,
в четырех байтах,в восьми байтах, в десяти байтах.

- задать любым трем числам произвольные метки, а в словах с именами ADR1,

ADR2, ADR3 определить адреса этих чисел. В сегменте данных любое число кроме последнего обозначить меткой MMM.

1. Тело программы должно содержать следующее:
   - в регистр CX непосредственно занести номер дома;
   - в регистр BX занести второе слово содержащееся за меткой MMM;
   - в регистр AX занести число или данное находящееся по адресу ADR2,используя косвенную адресацию;
   - считая что все эти данные числовые, получить их сумму в регистрахDX и AL;
   - полученную сумму занести в сегмент данных в байт LL и в слово XX;
   - выход в DOS.
2. Последовательно получить:

   файл с расширением .asm,файл с расширением .obj, файл с расширением .lst, файл с расширением .exe,

используя любой редактор и программы MASM.EXE, LINK.EXE

Задание 3.

Написать программу на языке программирования Ассемблер: сложит двачисла, находящиеся по адресу **pp** и **pp1,** результат занести по адресу **sum.**

Последовательно получить:

файл с расширением .asm,файл с расширением .obj, файл с расширением .lst, файл с расширением .exe.

Прорешать созданную программу в DEBUG, найти сегмент данных, сегменткодов, сегмент стека, связывая данные в листинге с данными в памяти

Отчет по листингу или в отладчике.

Изменить созданную программу, написанную на АССЕМБЛЕР'е таким образом, чтобы получить исходный модуль с расширением .COM.

Поставить файл с расширением .COM на выполнение.

Задание 4.

1. Составить программу на языке АССЕМБЛЕРА, задавая все определения сучетом того, что выполняемый модуль должен иметь расширение .EXE.

В сегменте данных любое число кроме последнего обозначить меткой MMM.

2. Тело программы должно содержать следующее:
   - в регистр CX непосредственно занести номер дома;
   - в регистр BX занести второе слово содержащееся за меткой MMM;
   - в регистр AX занести число или данное находящееся по адресу ADR2используя косвенную адресацию;
   - считая, что все эти данные числовые получить их сумму в регистреDX;
   - полученную сумму занести в сегмент данных;
   - конец.
3. Последовательно получить:

   файл с расширением .asm,файл с расширением .obj, файл с расширением .lst, файл с расширением .exe.

4. Прорешать созданную программу в DEBUG, найти сегмент данных, сегмент кодов, сегмент стека, связывая данные в листинге с данными в памяти.
5. Работать только в своей директории.
6. Отчет по листингу или в отладчике.
7. Изменить созданную программу, написанную на АССЕМБЛЕР'е таким образом, чтобы получить исходный модуль с расширением .COM
8. Поставить файл с расширением .COM на выполнение

Критерий оценки:
правильно выполнены 4 задания – оценка «5» правильно выполнены
3 задания – оценка «4»правильно выполнены 2 задания – оценка «3»

Самостоятельная работа №12

Название работы: создание программ на языке Ассемблер. **Цель:** проверка и корректировка текущих знаний студентов.**Уровень СРС:** воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).

Количество часов на выполнение: 2

Задание 1. Написать программу на языке Ассемблер, выбрав одну изпредложенных задач:

1. Ввести строку символьных данных, задавая буфер равный 18 байт.

Подсчитать в этой строке количество символов "i" .Выдать подсчитанное количество символов.

2. Ввести строку символьных данных, задавая буфер равный 10 байт.Проанализировать встречающиеся символы. Выдать одно из сообщений:

"Символы русского регистра"; "Символы латинского регистра";
"Символы и русского и латинского регистров".

3. Ввести строку символьных данных, задавая буфер равный 12 байт. Переставить символы в строке следующим образом: первый символ на место последнего, второй символ на место предпоследнего, предпоследний на место второго, а последний на место первого. Выдать полученную строку символов в 10 строку экрана, начиная с 30 позиции.

4. Ввести строку символьных данных, задавая буфер равный 18 байт. Разделить цепочку на две равные девяти байт каждая, выдать на экран эти части одна под другой начиная с 10 строки 35 столбца, в начале вторую часть, а затем первую.

5. Ввести строку символьных данных, задавая буфер равный 25 байт. Вводить только английские символы. Отсортировать строку по возрастаниюи выдать на экран в 4 строку.

6. Ввести строку символьных данных, задавая буфер равный 30 байт. Поменять в этой строке порядок символов: первый символ поставить на место второго, а второй на место первого, таким же образом переставить и все остальные символы: третий символ на место четвертого, а четвертый на место третьего и т. д. Выдать полученную строку символов в 7 строку экрана,начиная с 7 позиции.

7. Ввести одну из строк символьных данных: "единица", "два", "три", "четыре", "пять", "шесть", "семь", "восемь", "девять" – проанализировав введенные данные выдать на экран 1 или 2 или 3 или 4 или 5 или 6 или 7 или 8 или 9.

Критерий оценки:

правильно выполнено одно из заданий – оценка «5»
выполнено одно из заданий, но содержит ошибки – оценка «4» или «3»

Самостоятельная работа №13 Название работы:

создание программ с разветвлением.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).
Количество часов на выполнение: 2

Задание 1. Написать программу на языке программирования Ассемблер.
Дан ряд чисел. Если в характеристике XAR появляется определённое число, то числа из буфера складываются, в противоположном случае они вычитаются.
Задание 2. Составить программу обнуления памяти (любой области, заданной в сегменте данных) в размере 10 шестнадцатиразрядных слов.Задать буфер следующим образом:

buf db 20 dup('*') .

Критерий оценки:
правильно выполнены 2 задания – оценка «5» правильно выполнено 1 задание –
оценка «4» выполнены 2 задания, но содержат ошибки – оценка «3»

Самостоятельная работа №14-16

Название работы: создание программ с использованием циклов.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.
Форма контроля: письменная самостоятельная работа (в электронном виде).

Количество часов на выполнение: 6

Задание 1. Составить программу занесения в каждый байт буфера размером 25 байт числа - 0FCH.
Задание 2. Составить программу занесения в память последовательной цепочки чисел (1, 2, 3 и т.д. до 16), учитывая, что каждое число занимает 2 байта памяти.
Задание 3. Составить программу занесения в память последовательной цепочки чисел (0, 2, 4 и т.д. до 20), учитывая, что каждое число занимает 1 байт памяти.
Задание 4. Составить программу занесения в память последовательной цепочки чисел (100, 99, 98 и т.д. до 1), учитывая, что каждое число занимает2 байта памяти.
Задание 5. Составить программу занесения в память последовательной цепочки чисел (66, 64, 62 и т.д. до 0), учитывая, что каждое число занимает 1 байт памяти.

Критерий оценки:
правильно выполнены 5 заданий – оценка «5»правильно выполнены 4
задания – оценка «4»правильно выполнены 3 задания – оценка «3»

Самостоятельная работа №17

Название работы: создание программ с использованием логическихопераций.

Цель: проверка и корректировка текущих знаний студентов.
Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).

Количество часов на выполнение: 2

Задание 1

1. Составить программу на АССЕМБЛЕР'е, как программу с расширением

.exe, введя в качестве данных одно следующее чисел

11000011 10001111

2. Заполнить буфер памяти в байтах= (14 * 4) символами #, если бит предложенного слова равный14 в двоичном слове будет=1 и заполнить этот же буфер нулями, если рассмотренный бит =0.
3. Выделить этот бит и в качестве 1 или 0 записать в регистр dx.
4. Поместить за заполненным буфером два символа @@, если анализируемое число положительно, иначе поместить два символа &&.
5. Прорешать, созданную программу в DEBUG'e, меняя анализируемоечисло.

Критерий оценки:
правильно выполнено и скомпилировано 1 задание – оценка «5» частично выполнено и скомпилировано 1 задание – оценка «4» частично выполнено и нескомпилировано 1 задание – оценка «3»

Самостоятельная работа №18

Название работы: создание программ с использованием циклов и логических операций.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).
Количество часов на выполнение: 1

Задание. Изменить предыдущую программу (см. СРС №17) таким образом:

1. Если бит предложенного слова равный 14 в двоичном слове =1, то выдать сообщение "Бит равен единице"; если рассмотренный бит =0, то выдать сообщение "Бит равен нулю".
2. Выделить этот бит и в качестве 1 или 0 записать в регистр dx.
3. Проанализировать знак числа, если предложенное число >0 ,выдать сообщение "Число положительное", иначе выдать сообщение "Число

отрицательное".

4. Прорешать созданную программу в DOS.

Критерий оценки:
правильно выполнено и скомпилировано 1 задание – оценка «5» частично выполнено и скомпилировано 1 задание – оценка «4» частично выполнено и нескомпилировано 1 задание – оценка «3»

Самостоятельная работа №19

Название работы: создание программ с использованием команд сдвигов.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).

Количество часов на выполнение: 2

Задание:

```
CCC     SEGMENT
        assume      DS:CCC,CS:CCC,SS:CCCORG
                    100H
VX:     JMP         PP
FIF     DB          5
PP      PROC        NEAR
        MOV         AL,FIF
        SHR         AL,1
        MOV         AL,-5
        SHR         AL,1

        MOV         AL,FIF
        SHL         AL,1
        MOV         AL,-5
        SHL         AL,1

        MOV         AL,FIF
        SAR         AL,1
        MOV         AL,-5
        SAR         AL,1

        MOV         AL,FIF
        SAL         AL,1
        MOV         AL,-5
        SAL         AL,1

        RET
pp      ENDP
ccc     ENDS
        END         VX
```

Критерий оценки:

правильно выполнено и расписаны все сдвиги в 2-ой системе – оценка «5»правильно выполнено и нет объяснения сдвигов – оценка «4»

частично выполнено 1 задание – оценка «3»

Самостоятельная работа №20

Название работы: составление программ с использованием ввода-вывода наэкран.

Цель: проверка и корректировка текущих знаний студентов.

Уровень СРС: воспроизводящая.

Форма контроля: письменная самостоятельная работа (в электронном виде).

Количество часов на выполнение: 2

Задание 1.

Ввести строку символьных данных, задавая буфер равный 30 байт. Подсчитать в этой строке количество символов "f". Выдать одно изсообщений:

"Символа f в строке данных нет"; "Символ f встречается 1 раз";

"В строке данных символов f >=2".

Задание 2.

Ввести строку символьных данных, задавая буфер равный 40 байт. Заменить в этой строке встречающийся символ "d" на символ "s". Выдать полученную строку символов в первую строку экрана, начиная с 10 позиции.

Раздел 3

Основы программирования на процедурном алгоритмическом языке СИ

Самостоятельная работа №21

1. Задание. Изучить дополнительную литературу и составить информационное сообщение на темы:
«Язык программирования высокого уровня Си».
«Язык программирования С++»
«Популярные языки программирования для создания операционных систем»
2. Задание. Составить сравнительную таблицу достоинств и недостатков языков программирования
Студент должен:
уметь:
работать с литературой и выявлять главную часть, анализируя полученную информацию.
знать:
языки программирования высокого уровня, их отличия, основную область применения.
Критерии оценки подготовки сообщения:
- полнота и качественность информации по заданной теме;
- свободное владение материалом сообщения;
- логичность и четкость изложения материала;
- использование фактов при изложении материала, примеров, жизненных ситуаций;
- наличие и качество презентационного материала.

Пример выполнения

Задания 1. на тему: «Язык программирования высокого уровня»

Язык программирования высокого уровня - язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта языков программирования высокого уровня - это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания.

Так, языки программирования высокого уровня стремятся не только облегчить решение сложных программных задач, но и упростить портирование программного обеспечения. Использование разнообразных трансляторов и интерпретаторов обеспечивает связь программ, написанных при помощи языков высокого уровня, с различными операционными системами и оборудованием, в то время как их исходный код остаётся, в идеале, неизменным.

Такого рода оторванность высокоуровневых языков от аппаратной реализации компьютера помимо множества плюсов имеет и минусы. В частности, она не позволяет создавать простые и точные инструкции к используемому оборудованию. Программы, написанные на языках высокого уровня, проще для понимания программистом, но менее эффективны, чем их аналоги, создаваемые при помощи низкоуровневых языков. Одним из следствий этого стало добавление поддержки того или иного языка низкого уровня (язык ассемблера) в ряд современных профессиональных высокоуровневых языков программирования.

Примеры: C, C++, Visual Basic, Java, Python, PHP, Ruby, Perl, Delphi (Pascal). Языкам высокого уровня свойственно умение работать с комплексными структурами данных. В большинство из них интегрирована поддержка строковых типов, объектов, операций файлового ввода-вывода и т. п.

Первым языком программирования высокого уровня считается компьютерный язык Plankalkül разработанный немецким инженером Конрадом Цузе ещё в период 1942—1946 гг. Однако транслятора для него не существовало до 2000 г. Первым в мире транслятором языка высокого уровня является ПП (Программирующая Программа), он же ПП-1, успешно испытанный в 1954 г. Транслятор ПП-2 (1955 г., 4-й в мире транслятор) уже был оптимизирующим и содержал собственный загрузчик и отладчик, библиотеку стандартных процедур, а транслятор ПП для ЭВМ Стрела-4 уже содержал и компоновщик (linker) из модулей. Однако, широкое применение высокоуровневых языков началось с возникновением Фортран а и созданием компилятор а для этого языка (1957).

Пример выполнения

Задание 2. Составить сравнительную таблицу достоинств и недостатков языков программирования

Таблица 1

Языки программирования	Достоинства	Недостатки

Литература:

1. Фуфаев Э.В., Фуфаева Л.И.Пакеты прикладных программ: учеб. Пособие для студ. сред. проф. образования / под ред. Э.В. Фуфаев, Л.И. Фуфаева. – Москва: Издательский центр «Академия», 2014. – 352 с
2. Шилдт, Герберт. С++: базовый курс, 3-е издание. : Пер. с англ. – М.: Издательский дом «Вильямс», 2014. – 624с.: ил. – Парал. тит. англ.
3. Б. Керниган, Д. Ритчи, А. Фьюер. Язык программирования Си. Задачи по языку Си. М.: Финансы и статистика, 2013. – 564 с.
4. Энциклопедия. Режим доступа. URL: https://ru.wikipedia.org/wiki/ (дата обращения: 05.09.2016)
5. Bourabai Research. Режим доступа. URL: http://bourabai.ru/alg/classification04.htm (дата обращения: 05.09.2016)

Самостоятельная работа №22

1. Задание. Создать презентацию по темам:
«Алгоритмизация и требования к алгоритму»
«Основные этапы разработки программного приложения»
«Структурное программирование»
«Отладка и тестирование программы»
«Программное приложение C++ Builder»
2. Задание. Создать программное приложение арифметического выражения для заданных значений *x, y, z*.
Студент должен:
уметь:
- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;

знать:
- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно - ориентированного программирования;
- основные принципы отладки и тестирования программных продуктов;

Критерии оценки презентации
1. Содержательный критерий
правильный выбор темы, знание предмета и свободное владение текстом, грамотное использование научной терминологии, импровизация, речевой этикет
2. Логический критерий
стройное логико-композиционное построение речи, доказательность, аргументированность
3. Речевой критерий
использование языковых (метафоры, фразеологизмы, пословицы, поговорки и т.д.) и неязыковых (поза, манеры и пр.) средств выразительности; фонетическая организация речи, правильность ударения, четкая дикция, логические ударения и пр.
4. Психологический критерий
взаимодействие с аудиторией (прямая и обратная связь), знание и учет законов восприятия речи, использование различных приемов привлечения и активизации внимания
5. Критерий соблюдения дизайн-эргономических требований к компьютерной презентации
соблюдены требования к первому и последним слайдам, прослеживается обоснованная последовательность слайдов и информации на слайдах, необходимое и достаточное количество фото- и видеоматериалов, учет особенностей восприятия графической (иллюстративной) информации, корректное сочетание фона и графики, дизайн презентации не противоречит ее содержанию, грамотное соотнесение устного выступления и компьютерного сопровождения, общее впечатление от мультимедийной презентации
Критерии оценки практического задания:
- оценка «отлично», выставляется если программа работает, отчет выполненной работы оформлен полностью.
- оценка «хорошо», выставляется, если программа работает, отчет оформлен частично.
- оценка «удовлетворительно» выставляется, если программа работает, но имеются погрешности в коде или не все компоненты функционируют, отчет оформлен частично.
- оценка «неудовлетворительно» выставляется, если программа не работает, отчет не оформлен.

Пример выполнения
1. Задание по теме: «Алгоритмизация и требования к алгоритму»
Процессор электронно-вычислительной машины (ЭВМ) или персонального компьютера (ПК) — это ее "мозг", который умеет выполнять лишь простейшие команды. Для решения сложных задач обработки информации программист должен составить *алгоритм* — подробное описание последовательности арифметических и логических действий, расположенных в строгом логическом порядке и позволяющих решить конкретную задачу. Составление такого пошагового описания процесса решения задачи называется ее *алгоритмизацией.* Слово алгоритм, по существу, является синонимом таких слов, как способ, рецепт и т. п.
Требования, предъявляемые к алгоритму:
1. *Однозначность* — предлагаемые действия должны быть "понятны" компьютеру, а порядок исполнения этих действий должен быть единственно возможным, любая неопределенность или двусмысленность недопустимы.
2. *Массовость* — пригодность алгоритма для решения не только данной задачи, а множества родственных задач, относящихся к общему классу.
3. *Детерминированность* — повтор результата при повторе исходных данных.
4. *Корректность* — способность алгоритма давать правильные результаты решения задачи при различных исходных данных.
5. *Конечность* — решение задачи должно быть получено за конечное число шагов алгоритма, "зацикливание" недопустимо.
6. *Эффективность* — для успешного решения задачи должны использоваться ограниченные ресурсы конкретного компьютера (время работы процессора, объем оперативной памяти, быстродействие жесткого диска и др.).

Блок-схемы алгоритмов

1. Способы записи алгоритма

Разработка алгоритма решения задачи — сложный творческий процесс. Записать алгоритм в виде компьютерной программы без каких-либо предварительных рассуждений может только опытный программист при решении небольшой по объему, четко поставленной задачи. В реальной жизни такие задачи встречаются редко, поэтому обычно разработчик сначала продумывает алгоритм и записывает его в какой-либо удобной форме, а затем реализует алгоритм в виде программы.

При разработке сложных коммерческих программных продуктов часто алгоритмизацию выполняет один человек, а запись программы по имеющемуся алгоритму — другой. Следовательно, необходимо иметь такие способы записи алгоритмов, которые легко воспринимаются человеком, но являются достаточно строгими, чтобы их можно было впоследствии перевести на язык компьютера.

Существуют различные варианты записи алгоритмов. К основным относятся описательный и графический способы. *Описательным* называется алгоритм, составленный на естественном, в частности, математическом языке. *Графический* способ отличает компактная и наглядная форма записи в виде специальных графических знаков с указанием связи между ними.

2. Блок-схемы

При разработке программ рекомендуется использовать графический способ записи алгоритма в виде блок-схемы. *Блок-схема* — это графическое изображение алгоритма в виде плоских геометрических фигур (блоков), соединенных линиями. Внутри блока записывается действие, которое нужно выполнить, или условие, которое необходимо проверить. Блок-схема — стандартный способ записи алгоритма, существует государственный стандарт (ГОСТ), содержащий перечень правил построения блок-схем.

Основные блоки, которые используются при составлении графического алгоритма, изображены на рис. 1, где *а*— начало (конец) алгоритма; *б* — блок ввода/вывода; *в* — операционный блок; *г* — логический (условный) блок; *д* — цикл с параметром (для параметра цикла указывается его начальное и конечное значение, шаг равен единице). Конструкция "цикл с параметром" отдельно ГОСТом не регламентируется, однако приведенное обозначение (шестиугольник) разрешено использовать для различных целей, поэтому такое изображение цикла на блок-схеме часто встречается в литературе по программированию. Цикл с параметром будет подробно рассматриваться далее. В ГОСТ 19.701-90 (ИСО 5807-85), дата введения 01.01.1992 г., приводится еще один возможный вариант обозначений для циклов. Однако он представляется неудачным (рис. 1, *е—ж).*

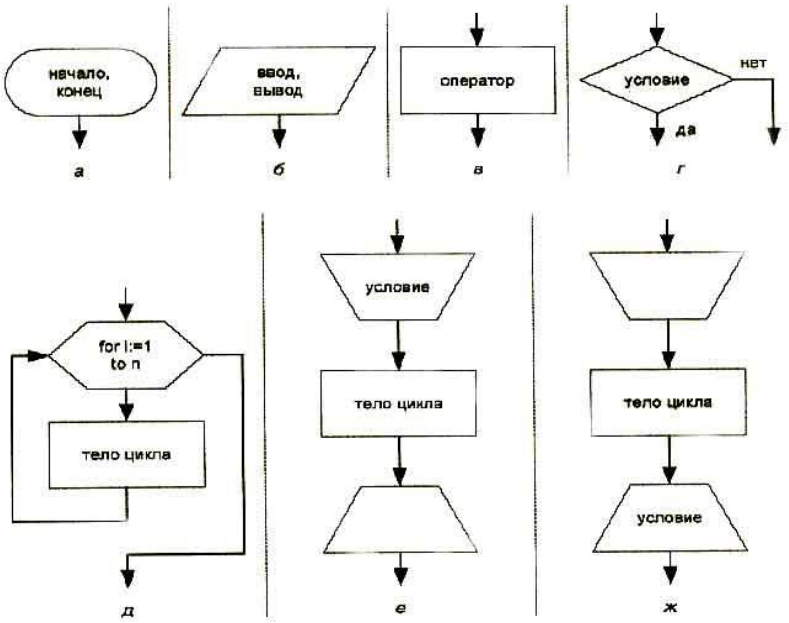


Рис.1. Условные обозначения на схемах алгоритмов

3. Следование, ветвление, цикл

Алгоритмические структуры (рис. 1, *а, б, в)* образуют линейную последовательность операций, которые выполняются по очереди в порядке записи, — *следование.* Программную реализацию такой алгоритмической структуры называют *линейной программой.* Линейные программы обычно предназначены для решения простейших задач, в которых не предусмотрен выбор из нескольких возможных направлений хода программы или циклическое повторение операций.

Возможность альтернативного выбора при выполнении программы предоставляют *ветвления* (рис.1, *г),* при выполнении которых алгоритм может пойти по одной из двух возможных ветвей в зависимости от справедливости проверяемого условия. Иногда выделяют также *обход,* который представляет собой пропуск нескольких шагов алгоритма при выполнении или невыполнении какого-либо условия.

Цикл (рис.1,*д)* представляет собой многократно повторяющуюся последовательность шагов алгоритма.

4.Пример блок-схемы алгоритма

Рассмотрим пример блок-схемы алгоритма игры "Угадай число".

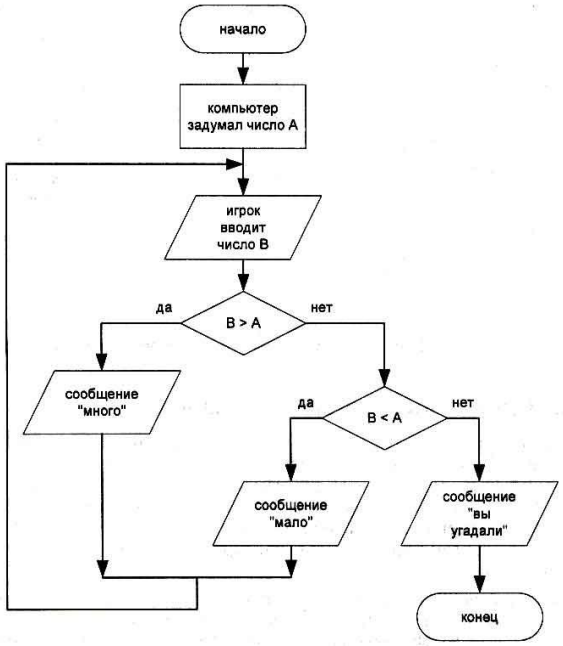


Рис.2. Блок-схема алгоритма игры "Угадай число"

Условие игры: игрок должен угадать число, "задуманное" компьютером — случайное число в диапазоне от 0 до 1000. Алгоритм игры представлен на рис. 2. Игра начинается с того, что компьютер задумывает случайное число A. В свою очередь, игрок вводит число B, пытаясь угадать значение A. Выполняется проверка: число B строго больше числа A? Если это так, т. е. "да", то компьютер выводит сообщение "много" и просит игрока ввести следующее число, еще раз испытав удачу. Если же результат — "нет", то компьютер выполняет следующую проверку: B строго меньше A? Если ее результат — "да", то выводится сообщение "мало", и компьютер ожидает ввода игроком следующего числа. Если же результат второй проверки — "нет", это означает, что игрок угадал число, задуманное компьютером. В таком случае выводится сообщение "вы угадали", и игра завершается.

Пример выполнения

2. Задание. Создать программное приложение арифметического выражения для заданных значений *x*, *y*, *z*.

$$u = tg^2(x+y) - e^{y-z}\sqrt{\cos x^2 + \sin z^2}$$

Пример создания оконного приложения

В оконном режиме панель диалога программы создать в виде, представленном на рис.2.

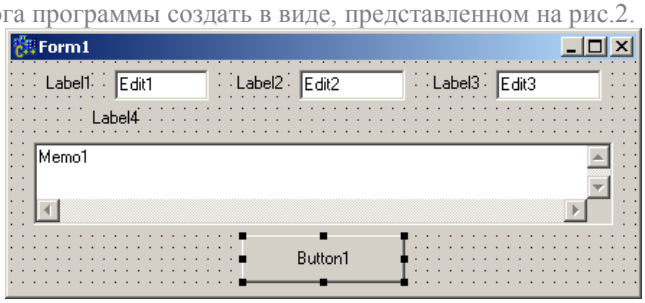


Рис. 2

Для создания проекта необходимо выполнить следующие действия.

1. Запускаем C++ *Builder*. Создаем папку, открыв ее, сохраняем предлагаемые файлы *Unit*1.*cpp* и *Project*1.*cpp* (рекомендуем без изменения).

2. Оформляем окно формы, заменив заголовок *Form*1 на нужный текст. Помещаем на форму необходимые компоненты *Label*1, *Label*2, *Label*3, *Label*4 (вставляя в *Caption* соответствующие тексты), *Edit*1, *Edit*2, *Edit*3, *Memo*1 с полосами прокрутки, *Button*1 (заменив в *Caption* текст).

Используя свойство *Font*, выбираем стили выводимых текстов.

3. Оформляем листинг программы (*Unit*1.*cpp*). Двойным щелчком кнопкой мыши по свободному месту формы создаем функцию *FormCreate* и заполняем. Переходим на форму (*F*12), щелкаем дважды по кнопке «ВЫПОЛНИТЬ» и заполняем созданную функцию *Button*1*Click*.

4. Перед запуском программы на обработку, ***сохраняем все.***

5. Запускаем проект на выполнение, исправляем ошибки.

Текст программы может иметь следующий вид (наклонным мелким шрифтом выделен текст, редактировать который не рекомендуется):

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
```

```
#include "Unit1.h"
#include "math.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Text = "3,4";
Edit2->Text = "7,4e-2";
Edit3->Text = "1,943e2";
Memo1->Clear();
Memo1->Lines->Add("Арифметическое выражение");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
double x, y, z, a, b ,c, rez;
x = StrToFloat(Edit1->Text);
y = StrToFloat(Edit2->Text);
z = StrToFloat(Edit3->Text);
a = pow(tan(x+y),2);
b = exp(y-z);
c = sqrt(cos(x*x)+sin(z*z));
rez = a-b*c;
Memo1->Lines->Add("При x = "+FloatToStrF(x,ffFixed,7,3)
                                                    + "; y = "+FloatToStrF(y,ffFixed,7,3)+"; z = "+FloatToStrF(z,ffFixed,7,3));
Memo1->Lines->Add("Результат = "+FloatToStr(rez));
}
```

В строковых константах разделитель целой и дробной частей – запятая: Edit1->Text = "3,4"; в отличие от числовых констант в тексте программы.

В результате должно получиться рабочее окно (рис.3). Если щелкнуть мышью по кнопке «*ВЫПОЛНИТЬ*», в окне *Memo*1 появится соответствующий текст (результат). Далее в окошках *Edit** можно изменять исходные значения и, нажимая кнопку «*ВЫПОЛНИТЬ*», получать новые результаты.

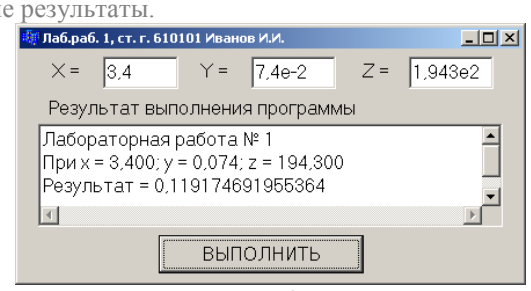


Рис.3

Создание консольного приложения

Чтобы создать проект в консольном приложении, выполняем следующую последовательность действий: *File* → *Close All* → *File* → *New* → *Other* → *Console Wizard* → *Ok*. Закрываем все окошки, кроме 5, которое в консольном приложении будет иметь вид

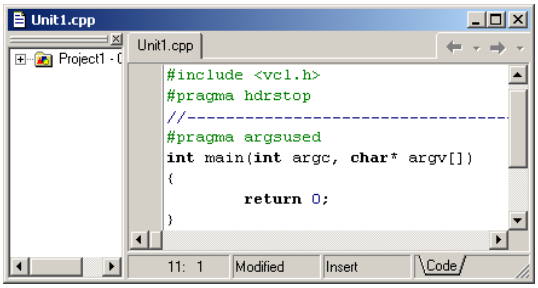


Рис. 4

Текст программы может иметь следующий вид:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
double x, y, z, a, b ,c, rez;
puts("\n\tx,y,z = ");
scanf("%lf%lf%lf", &x, &y, &z);
a = pow(tan(x+y),2);
b = exp(y-z);
c = sqrt(cos(x*x)+sin(z*z));
rez = a-b*c;
printf("\n x = %7.3lf\n y = %7.3lf\n z = %7.3lf\nRezult = %lf\n", x, y, z, rez);
puts("Press any key ... ");
getch();
return 0;
}
```

Для исходных данных x = 3,4; y = 7,4·10^{-2}; z = 1,943·10^{2}, результат выполнения программы выглядит следующим образом:

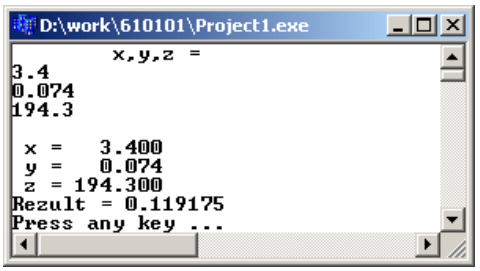


Рис. 5

Индивидуальные задачи к заданию 2.

Создать программу вычисления указанной величины. Результат проверить при заданных исходных значениях.

1. $t = \dfrac{2\cos(x-\pi/6)}{0{,}5+\sin^2 y}\left(1+\dfrac{z^2}{3-z^2/5}\right)$.
   При x = 14.26, y = –1.22, z = 3.5×10^{-2} : **0.564846.**
2. $u = \dfrac{\sqrt[3]{8+|x-y|^2+1}}{x^2+y^2+2} - e^{|x-y|}(tg^2 z+1)^x$.
   При x = –4.5, y = 0.75×10^{-4}, z = 0.845×10^{2} : **–55.6848.**
3. $v = \dfrac{1+\sin^2(x+y)}{\left|x-\dfrac{2y}{1+x^2y^2}\right|}x^{|y|} + \cos^2[arctg(1/z)]$.
   При x = 3.74×10^{-2}, y = –0.825, z = 0.16×10^{2} : **1.0553.**
4. $w = |\cos x - \cos y|^{(1+2\sin^2 y)}\left(1+z+\dfrac{z^2}{2}+\dfrac{z^3}{3}+\dfrac{z^4}{4}\right)$.
   При x = 0.4×10^{4}, y = –0.875, z = –0.475×10^{-3} : **1.9873**.
5. $\alpha = \ln(y^{-\sqrt{|x|}})(x-y/2)+\sin^2 arctg(z).$
   При x = –15.246, y = 4.642×10^{-2}, z = 20.001×10^{2} : **–182.036**.
6. $\beta = \sqrt{10(\sqrt[3]{x}+x^{y+2})}\cdot(\arcsin^2 z-|x-y|)$
   При x = 16.55×10^{-3}, y = –2.75, z = 0.15 : **40.630694**.
7. $\gamma = 5arctg(x)-\dfrac{1}{4}\arccos(x)\dfrac{x+3|x-y|+x^2}{|x-y|z+x^2}$.
   При x = 0.1722, y = 6.33, z = 3.25×10^{-4} : **–205.305571**.
8. $\varphi = \dfrac{e^{|x-y|}|x-y|^{x+y}}{arctg\ x+arctg\ z}+\sqrt[3]{x^6+\ln^2 y}.$
   При x = –2.235×10^{-2}, y = 2.23, z = 15.221 : **39.374**.
9. $\psi = |x^{y/x}-\sqrt[3]{y/x}|+(y-x)\dfrac{\cos y - z/(y-x)}{1+(y-x)^2}$.
   При x = 1.825×10^{2}, y = 18.225, z = –3.298×10^{-2} : **1.2131**.
10. $a = 2^{-x}\sqrt{x+\sqrt[4]{|y|}}\ \sqrt[3]{e^{x-1/\sin z}}$.
    При x = 3.981×10^{-2}, y = –1.625 × ×10^{3}, z = 0.512 : **1.26185**.

11. $b = y^{\sqrt[3]{|x|}} + \cos^3 y \dfrac{|x-y| \cdot \left(1 + \dfrac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|} + x/2}.$

При $x = 6.251$, $y = 0.827$, $z = 25.001$: **0.7121**.

12. $c = 2^{y^x} + (3^x)^y - \dfrac{y \cdot (arctgz - \pi/6)}{|x| + \dfrac{1}{y^2+1}}.$

При $x = 3.251$, $y = 0.325$, $z = 0.466 \times 10^{-4}$: **4.251433**.

13. $f = \dfrac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x-y|(\sin^2 z + tg\ z)}.$

При $x = 17.421$, $y = 10.365 \times 10^{-3}$, $z = 0.828 \times 10^{5}$: **0.33056**.

14. $g = \dfrac{y^{x+1}}{\sqrt[3]{|y-2|+3}} + \dfrac{x + y/2}{2|x+y|}(x+1)^{-1/\sin z}.$

При $x = 12.3 \times 10^{-1}$, $y = 15.4$, $z = 0.252 \times 10^{3}$: **82.825623**.

15. $h = \dfrac{x^{y+1} + e^{y-1}}{1 + x|y - tgz|}(1 + |y-x|) + \dfrac{|y-x|^2}{2} - \dfrac{|y-x|^3}{3}.$

При $x = 2.444$, $y = 0.869 \times 10^{-2}$, $z = -0.13 \times 10^{3}$: **–0.49871**.

16. $w = \sqrt[3]{x^6 + \ln^2 y} + \dfrac{e^{|x-y|}|x-y|^{x+y}}{arctg(x) + arctg(z)}.$

При $x = -2.235 \times 10^{-2}$, $y = 2.23$, $z = 15.221$: **39.374**.

Самостоятельная работа №23

1. Задание. Написать и отладить программу вывода всех значений функции $S(x)$ для аргумента x, изменяющегося в интервале от a до b с шагом h и заданном n.

Студент должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;

знать:

- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно - ориентированного программирования;
- основные принципы отладки и тестирования программных продуктов;

Критерии оценки:

- оценка «отлично», выставляется если программа работает, отчет выполненной работы оформлен полностью.
- оценка «хорошо», выставляется, если программа работает, отчет оформлен частично.
- оценка «удовлетворительно» выставляется, если программа работает, но имеются погрешности в коде или не все компоненты функционируют, отчет оформлен частично.
- оценка «неудовлетворительно» выставляется, если программа не работает, отчет не оформлен.

Пример выполнения

1. Задание. Написать и отладить программу вывода всех значений функции $S(x)$ для аргумента x, изменяющегося в интервале от a до b с шагом h и заданном n.

$$S(x) = \sum_{k=0}^{N} (-1)^k \frac{x^k}{k!}.$$

Панель диалога и полученные результаты представлены на рис.1.

Пример создания оконного приложения

Текст функций-обработчиков может быть следующим (стандартный текст опущен):

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
      Edit1->Text="0,1";      Edit2->Text="1,0";
      Edit3->Text="10";       Edit4->Text="0,2";
      Memo1->Lines->Add("Значение функции");
}
```

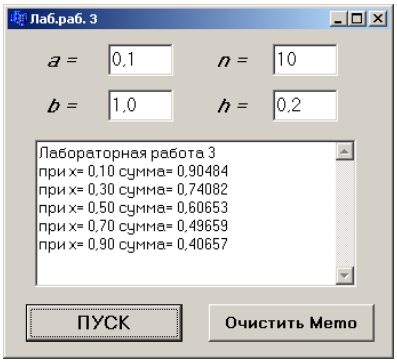


Рис.1

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
double a, b, x, h, r, s;
int n, zn = -1, k;
                        a = StrToFloat(Edit1->Text);
                        b = StrToFloat(Edit2->Text);
                        n = StrToInt(Edit3->Text);
                        h = StrToFloat(Edit4->Text);
            for(x = a; x<=b; x+=h) {
             r = s = 1;
             for(k = 1; k<=n; k++) {
                    r = zn*r*x/k;
                    s+=r;
             }
             Memo1->Lines->Add("при x= "+FloatToStrF(x,ffFixed,8,2)
                    +" сумма= "+FloatToStrF(s,ffFixed,8,5));
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
      Memo1->Clear();
}
```

Пример создания консольного приложения

Текст программы предложенного задания может иметь вид

```
#include <vcl.h>
#include <stdio.h>
#include <conio.h>
#pragma hdrstop
#pragma argsused
int main(int argc, char* argv[])
{
      double a, b, x, h, r, s;
      int n, zn = -1, k;
      puts("Input a,b,h,n");
                scanf("%lf%lf%lf%d", &a, &b, &h, &n);
      for(x = a; x<=b; x+=h) {
            r = s = 1;
            for(k = 1; k<=n; k++) {
                  r=zn*r*x/k;
                  s+=r;
            }
            printf("\n x= %8.2lf   sum= %8.5lf", x,s);
      }
      puts("\nPress any key ... ");
      getch();
      return 0;
}
```

Результат программы с введенными значениями a=0.1, b=1.0, h=0.2 и n=10:

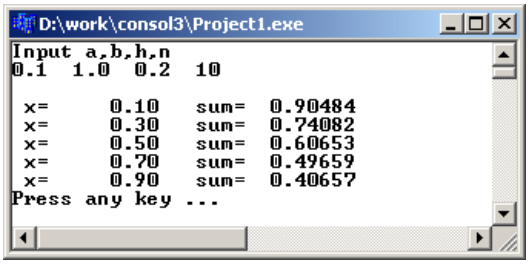


Рис. 2

Индивидуальные задачи к заданию 1.

Для каждого ***x***, изменяющегося от ***a*** до ***b*** с шагом ***h***, найти значения функции $Y(x)$, суммы $S(x)$ и $|Y(x)–S(x)|$ и вывести в виде таблицы. Значения a, b, h и n вводятся с клавиатуры. Так как значение $S(x)$ является рядом разложения функции $Y(x)$, при правильном решении значения S и Y для заданного аргумента x (для тестовых значений исходных данных) должны совпадать в целой части и в первых двух-четырех позициях после десятичной точки.

Работу программы проверить для $a = 0{,}1$; $b = 1{,}0$; $h = 0{,}1$; значение параметра n выбрать в зависимости от задания.

1. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k+1}}{(2k+1)!}$,	$Y(x)=\sin(x)$.
2. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}$,	$Y(x)=x\cdot arctg(x)-\ln\sqrt{1+x^2}$.
3. $S(x)=\sum_{k=0}^{n}\frac{\cos(k\pi/4)}{k!}x^k$,	$Y(x)=e^{x\cos\frac{\pi}{4}}\cos(x\sin(\pi/4))$
4. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{x^{2k}}{(2k)!}$,	$Y(x)=\cos(x)$.
5. $S(x)=\sum_{k=0}^{n}\frac{\cos(kx)}{k!}$,	$Y(x)=e^{\cos x}\cos(\sin(x))$.
6. $S(x)=\sum_{k=0}^{n}\frac{2k+1}{k!}x^{2k}$,	$Y(x)=(1+2x^2)e^{x^2}$.
7. $S(x)=\sum_{k=1}^{n}\frac{x^k\cos(k\pi/3)}{k}$,	$Y(x)=-\frac{1}{2}\ln(1-2x\cos\frac{\pi}{3}+x^2)$.
8. $S(x)=\sum_{k=0}^{n}\frac{(2x)^k}{k!}$,	$Y(x)=e^{2x}$.
9. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k+1}}{4k^2-1}$,	$Y(x)=\frac{1+x^2}{2}arctg(x)-x/2$.
10. $S(x)=\sum_{k=0}^{n}\frac{x^{2k}}{(2k)!}$,	$Y(x)=\frac{e^x+e^{-x}}{2}$.
11. $S(x)=\sum_{k=0}^{n}\frac{k^2+1}{k!}(x/2)^k$,	$Y(x)=(x^2/4+x/2+1)e^{x/2}$.
12. $S(x)=\sum_{k=0}^{n}(-1)^k\frac{2k^2+1}{(2k)!}x^{2k}$,	$Y(x)=(1-\frac{x^2}{2})\cos(x)-\frac{x}{2}\sin(x)$
13. $S(x)=\sum_{k=1}^{n}(-1)^k\frac{(2x)^{2k}}{(2k)!}$,	$Y(x)=2(\cos^2 x-1)$.
14. $S(x)=\sum_{k=0}^{n}\frac{x^{2k+1}}{(2k+1)!}$,	$Y(x)=\frac{e^x-e^{-x}}{2}$.
15. $S(x)=\sum_{k=1}^{n}(-1)^{k+1}\frac{x^{2k}}{2k(2k-1)}$,	$Y(x)=-\ln\sqrt{1+x^2}+x\,arctg(x)$
16. $S(x)=\sum_{k=0}^{n}\frac{\cos(k\pi/4)}{k!}x^k$,	$Y(x)=\cos[x\cdot\sin(\pi/4)]e^{x\cos\frac{\pi}{4}}$.

Литература:

1. Фуфаев Э.В., Фуфаева Л.И.Пакеты прикладных программ: учеб. Пособие для студ. сред. проф. образования / под ред. Э.В. Фуфаев, Л.И. Фуфаева. – Москва: Издательский центр «Академия», 2014. – 352 с

2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. – М.: Издательский дом «Вильямс», 2014. – 624с.: ил. – Парал. тит. англ.

3. Б. Керниган, Д. Ритчи, А. Фьюер. Язык программирования Си. Задачи по языку Си. М.: Финансы и статистика, 2015. – 564 с.

Самостоятельная работа №24

«Проверка состава оборудования»

1. Задание. Написать и отладить программу обработка одномерных массивов

Студент должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- выполнять отладку и тестирование программы на уровне модуля;

знать:

- основные этапы разработки программного обеспечения;
- основные принципы отладки и тестирования программных продуктов;

Критерии оценки:

- оценка «отлично», выставляется если программа работает, отчет выполненной работы оформлен полностью.
- оценка «хорошо», выставляется, если программа работает, отчет оформлен частично.
- оценка «удовлетворительно» выставляется, если программа работает, но имеются погрешности в коде или не все компоненты функционируют, отчет оформлен частично.
- оценка «неудовлетворительно» выставляется, если программа не работает, отчет не оформлен.

Пример выполнения

1. Задание. Написать и отладить программу обработка одномерных массивов.

Значение *N* вводить из *Edit*, значения массива *A* – из компоненты *StringGrid*. Результат вывести в компоненту *StringGrid*.

Панель диалога и результаты выполнения программы приведена на рис. 1.

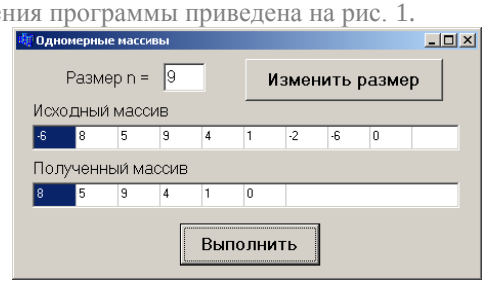


Рис.1

Настройка компоненты StringGrid

На закладке *Additional* выберите пиктограмму  , установите компоненты *StringGrid*1 и *StringGrid*2 и отрегулируйте их размеры. В инспекторе объектов для обоих компонент установите значения *ColCount* равными 2, *RowCount* равными 1, т.е. по два столбца и одной строке, а значения *FixedCols* и *FixedRows* равными 0. Значение ширины клетки столбца *DefaultColWidth* равным 40.

По умолчанию в компоненту *StringGrid* ввод данных разрешен только программно. Для разрешения ввода данных с клавиатуры необходимо в свойстве ***Options*** строку ***goEditing*** для компоненты *StringGrid*1 установить в положение ***true***.

Текст функций-обработчиков может иметь следующий вид:

```
        . . .
int n = 4;
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    randomize();                    // Изменение начального адреса для random()
    Edit1->Text=IntToStr(n);
    StringGrid1->ColCount=n;
    for(int i=0; i<n;i++)           // Заполнение массива A случайными числами
        StringGrid1->Cells[i][0] = IntToStr(random(21)-10);
    Label3->Hide();                 // Скрыть компоненту
    StringGrid2->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    n=StrToInt(Edit1->Text);
    if(n>10){
        ShowMessage("Максимальное количество 10!");
        n=10;
        Edit1->Text = "10";
    }
    StringGrid1->ColCount=n;
    for(int i=0; i<n;i++)
```

```
            StringGrid1->Cells[i][0]=IntToStr(random(21)-10);
        Label3->Hide();
        StringGrid2->Hide();
    }
    //---------------------------------------------------------------------------
    void __fastcall TForm1::Button2Click(TObject *Sender)
    {
        int i, kol = 0, a[10];                  // Декларация одномерного массива
    //Заполнение массива A элементами из таблицы StringGrid1
        for(i=0; i<n;i++)
            a[i]=StrToInt(StringGrid1->Cells[i][0]);
    //Удаление отрицательных элементов из массива A
        for(i=0; i<n;i++)
            if(a[i]>=0) a[kol++] = a[i];
        StringGrid2->ColCount = kol;
        StringGrid2->Show();                    // Показать компоненту
        Label3->Show();
    //Вывод результата в таблицу StringGrid2
        for(i=0; i<kol;i++) StringGrid2->Cells[i][0]=IntToStr(a[i]);
    }
```

Пример создания консольного приложения

Текст программы может иметь следующий вид (обратите внимание на то, что функция *main* используется в простейшей форме – без параметров и не возвращает результатов):

```
    . . .
#include <stdio.h>
#include <conio.h>
void main()
{
            int a[10],n, i, kol=0;
   randomize();                                 // Изменение начального адреса для random()
    printf("Input N (<=10) ");
    scanf("%d", &n);
    puts("\n Massiv A");
    for(i=0; i<n;i++) {
        a[i] = random(21)-10;          // Заполнение массива A случайными числами
        printf("%4d", a[i]);
    }
//Удаление отрицательных элементов из массива A
    for(i=0; i<n;i++)
        if(a[i]>=0)  a[kol++] = a[i];
    puts("\n Rezult massiv A");
    for(i=0; i<kol;i++) printf("%4d", a[i]);
    puts("\n Press any key ... ");
    getch();
}
```

С заполненным случайными числами массивом *A* результат программы может быть следующим:

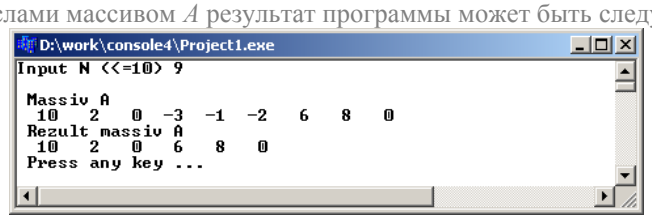


Рис. 2

Индивидуальные задачи к заданию 1.

Написать программу по обработке одномерных массивов. Размеры массивов вводить с клавиатуры. В консольном приложении предусмотреть возможность ввода данных как с клавиатуры, так и с использованием функции *random*().

При создании оконного приложения скалярный (простой) результат выводить в виде компоненты *Label*, а массивы вводить и выводить с помощью компонент *StringGrid*.

В одномерном массиве, состоящем из *n* вводимых с клавиатуры целых элементов, вычислить:

Произведение элементов массива, расположенных между максимальным и минимальным элементами.

Сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Сумму элементов массива, расположенных до последнего положительного элемента.

Сумму элементов массива, расположенных между первым и последним положительными элементами.

Произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Сумму элементов массива, расположенных между первым и вторым отрицательными элементами.
Сумму элементов массива, расположенных до минимального элемента.
Сумму модулей элементов массива, расположенных после последнего отрицательного элемента.
Сумму элементов массива, расположенных после последнего элемента, равного нулю.
Сумму модулей элементов массива, расположенных после минимального по модулю элемента.
Сумму элементов массива, расположенных после минимального элемента.
Сумму элементов массива, расположенных после первого положительного элемента.
Сумму модулей элементов массива, расположенных после первого отрицательного элемента.
Сумму модулей элементов массива, расположенных после первого элемента, равного нулю.
Сумму положительных элементов массива, расположенных до максимального элемента.
Произведение элементов массива, расположенных между первым и последним отрицательными элементами.

Самостоятельная работа №25 . « Функция таймера – звук и время»

1. Задание: Написать и отладить программу обработка двухмерных динамических массивов

Студент должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- выполнять отладку и тестирование программы на уровне модуля;

знать:

- основные этапы разработки программного обеспечения;
- основные принципы отладки и тестирования программных продуктов;

Критерии оценки:

- оценка «отлично», выставляется если программа работает, отчет выполненной работы оформлен полностью.
- оценка «хорошо», выставляется, если программа работает, отчет оформлен частично.
- оценка «удовлетворительно» выставляется, если программа работает, но имеются погрешности в коде или не все компоненты функционируют, отчет оформлен частично.
- оценка «неудовлетворительно» выставляется, если программа не работает, отчет не оформлен.

Пример выполнения

1. Задание: Написать и отладить программу обработка двухмерных динамических массивов

Рассчитать значения вектора $\vec{Y} = A \cdot \vec{B}$, где *A* – квадратная матрица размером *N*×*N*, а *Y* и *B* – одномерные массивы размером *N*. Элементы вектора *Y* определяются по формуле

$$Y_i = \sum_{j=0}^{N-1} A_{ij} \cdot B_j .$$

Значение *N* вводить из *Edit*, *A* и *B* – из компонент *StringGrid*. Результат вывести в компоненту *StringGrid*.

Панель диалога и результаты выполнения программы приведена на рис. 1.

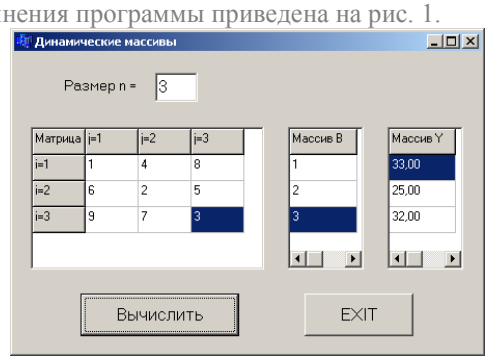


Рис. 1

Настройка компонент StringGrid

Для компоненты *StringGrid*1 значения *ColCount* и *RowCount* установите равными, например, 3 – три столбца и три строки, а *FixedCols* и *FixedRows* – 1.

Так как компоненты *StringGrid*2 и *StringGrid*3 имеют только один столбец, то у них *ColCount* = 1, *RowCount* = 3, а *FixedCols* = 0 и *FixedRows* = 1.

В свойстве ***Options*** строку ***goEditing*** для компонент *StringGrid*1 и *StringGrid*2 установите в положение ***true***.

Для изменения размера *n* используется функция-обработчик *EditChange*, полученная двойным щелчком по компоненте *Edit*.

Текст программы может иметь следующий вид:

```
                                    . . .
//---------------------- Глобальные переменные  -------------------
      int n = 3;
      double **a, *b;                                          // Декларации указателей
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
        Edit1->Text=IntToStr(n);
        StringGrid1->ColCount = n+1;        StringGrid1->RowCount = n+1;
        StringGrid2->RowCount = n+1;       StringGrid3->RowCount = n+1;
// Ввод в левую верхнюю ячейку таблицы названия массивов
        StringGrid1->Cells[0][0] = "Матрица A";
        StringGrid2->Cells[0][0] = "Массив B";
        StringGrid3->Cells[0][0] = "Массив Y";
        for(int i=1; i<=n;i++){
         StringGrid1->Cells[0][i]="i="+IntToStr(i);
         StringGrid1->Cells[i][0]="j="+IntToStr(i);
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
        int i;
        n=StrToInt(Edit1->Text);
        StringGrid1->ColCount = n+1;        StringGrid1->RowCount = n+1;
        StringGrid2->RowCount = n+1;        StringGrid3->RowCount = n+1;
        for(i=1; i<=n;i++){
                StringGrid1->Cells[0][i]="i="+IntToStr(i);
                StringGrid1->Cells[i][0]="j="+IntToStr(i);
        }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        double s;
        int i,j;
        a = new double*[n];                                        // Захват памяти под указатели
        for(i=0; i<n;i++)  a[i] = new double[n];              // Захват памяти под элементы
        b = new double[n];
// Заполнение массивов A и B элементами из таблиц StringGrid1 и StringGrid2
                for(i=0; i<n;i++) {
                                for(j=0; j<n;j++)   a[i][j]=StrToFloat(StringGrid1->Cells[j+1][i+1]);
                                b[i]=StrToFloat(StringGrid2->Cells[0][i+1]);
                }
// Умножение строки матрицы A на вектор B и вывод результата s в StringGrid3
                for(i=0; i<n;i++){
                                for(s=0, j=0; j<n;j++)  s += a[i][j]*b[j];
                                StringGrid3->Cells[0][i+1] = FloatToStrF(s, ffFixed,8,2);
                }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
     delete []a;
     delete []b;
     ShowMessage("Память освобождена!");
     Close();
}
```

Пример создания консольного приложения

Текст программы может иметь следующий вид:

```
                                        . . .
void main()
{
    double **a, *b, s;
    int i, j, n;
            printf(" Input size N : ");    scanf("%d",&n);
a = new double*[n];                                        // Захват памяти под указатели
for(i=0; i<n;i++)
                        a[i] = new double[n];                                // Захват памяти под элементы
```

```
b = new double[n];
            puts("\n Input Massiv A:");
for(i=0; i<n;i++)
         for(j=0; j<n;j++)    scanf("%lf", &a[i][j]);
puts("\n Input Massiv B:");
for( i=0; i<n;i++)       scanf("%lf", &b[i]);
puts("\n Massiv Y:");
for(i=0; i<n;i++){
        for(s=0, j=0; j<n;j++)  s+=a[i][j]*b[j];
            printf("  %8.2lf ", s);
}
delete []a;
delete []b;
puts("\n Delete !");
puts("\n Press any key ... ");
getch();
}
```

При вводе значений элементов массивов в одной строке через пробелы должен получиться следующий результат:

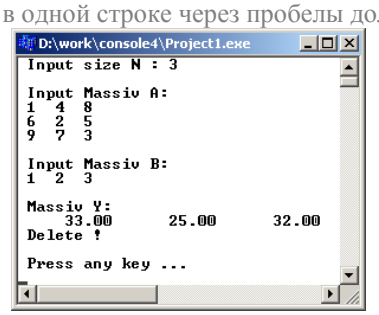


Рис. 2.

Индивидуальные задачи к заданию 1.

Написать программу по обработке динамических массивов. Размеры массивов вводить с клавиатуры. При создании оконного приложения скалярный (простой) результат выводить в виде компоненты *Label*, а массивы вводить и выводить с помощью компонент *StringGrid*, в которых 0-й столбец и 0-ю строку использовать для отображения индексов массивов.

1. Из матрицы размером $N \times M$ получить вектор B, присвоив его k-му элементу значение 0, если все элементы k-го столбца матрицы нулевые, иначе 1.

2. Из матрицы размером $N \times M$ получить вектор B, присвоив его k-му элементу значение 1, если элементы k-й строки матрицы упорядочены по убыванию, иначе 0.

3. Из матрицы размером $N \times M$ получить вектор B, присвоив его k-му элементу значение 1, если k-я строка матрицы симметрична, иначе значение 0.

4. Задана матрица размером $N \times M$. Определить количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером $N \times M$. Определить количество элементов матрицы, у которых слева находится элемент больше его, а справа – меньше.

6. Задана матрица размером $N \times M$. Определить количество различных значений матрицы, т.е. повторяющиеся элементы считать один раз.

7. В матрице размером $N \times M$ упорядочить строки по возрастанию их первых элементов.

8. В матрице размером $N \times M$ упорядочить строки по возрастанию суммы их элементов.

9. В матрице размером $N \times M$ упорядочить строки по возрастанию их наибольших элементов.

10. Определить, является ли квадратная матрица симметричной относительно побочной диагонали.

11. Задана матрица размером $N \times M$. Определить количество элементов матрицы, у которых слева находится элемент меньше его, а справа – больше.

12. В квадратной матрице найти произведение элементов, лежащих выше побочной диагонали.

13. В квадратной матрице найти максимальный среди элементов, лежащих ниже побочной диагонали.

14. В матрице размером $N \times M$ поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим значением.

15. Из матрицы размером n получить матрицу размером n-1 путем удаления строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.

16. В матрице размером n найти сумму элементов, лежащих ниже главной диагонали, и произведение элементов, лежащих выше главной диагонали.

Самостоятельная работа №26

. «Клавиатура»

1. Задание: Изучить дополнительную литературу и подготовить информационное сообщение на тему:

«Распределение памяти»

«Аппаратные прерывания»

«Управление видеоадаптером»

«Распределение памяти, логическая структура диска и физический дисковый адрес»

Студент должен:

- **знать,** о загрузке системы, распределение памяти, о аппаратном прерывании и скан-кодах, драйверах и переопределении клавиатуры, типы видеоадаптеров, видеопамять, режимы видеоадаптера, о внешней памяти, физический дисковый адрес, логическая структура диска

- **уметь**, работать с литературой и выявлять главную часть, анализируя полученную информацию.

Критерии оценки подготовки сообщения:

- полнота и качественность информации по заданной теме;
- свободное владение материалом сообщения;
- логичность и четкость изложения материала;
- использование фактов при изложении материала, примеров, жизненных ситуаций;
- наличие и качество презентационного материала.

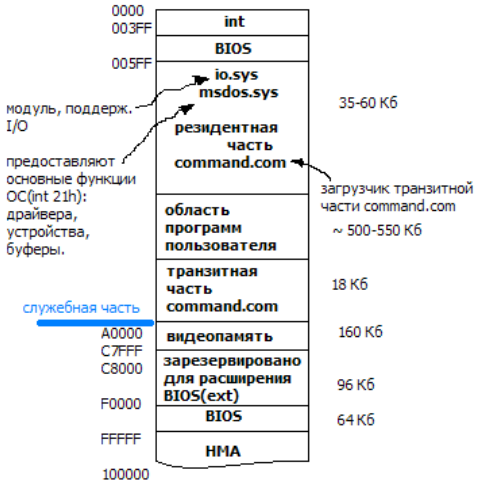
Пример выполнения

1. Задание на тему: Распределение памяти

MS-DOS

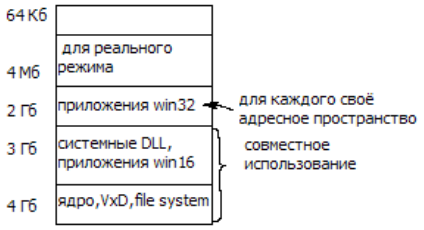
Вся память может быть разбита на 3 части:

1. Таблица векторов прерываний (1 Кб) + 512 Кб для размещения глобальных переменных BIOS;
2. ОС;
3. Пользовательские программы.



i8086 и 8088 могли адресовать 1 Мб памяти, с i80286 адресовалось до 16 Мб памяти (после добавления в BIOS HMA – High Memory Area), MS-DOS использует HMA ограничено – туда можно загрузить драйвер, часть самой ОС и буферы для ОС. MS-DOS 6.2.2 была уже достаточно большой (большинство модулей загружались в HMA).

Windows 95/98

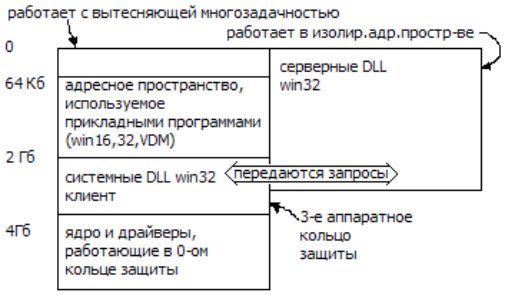


Для загрузки используется MS-DOS 7.0. В обычном режиме после загрузки процессор переключается в защищённый режим и начинает распределять память с помощью страничного механизма. Загружается GUI (Graphic User Interface - графический пользовательский интерфейс). Если GUI не загружать, то запустится MS-DOS. В защищённом режиме используется плоская модель памяти, при который все сегменты совпадают друг с другом и имеют максимально возможный размер. Любая прикладная программа оперирует с 32-разрядными адресами. Сегмент кода совпадает с сегментом данных. Виртуальное адресное пространство делится на страницы по 4 Кб. В младших адресах виртуального адресного пространства находятся драйвера реального режима, там находятся резидентные программы. В Win 95 программа выполняется в своём адресном пространстве, но все приложения используют один и тот же 32-разрядный системный код, который расположен между 2 и 4 Гб.

Можно выполнять системные вызовы в адресном пространстве программы, что заметно повышает быстродействие, но при этом возможен доступ к этой области и повреждение системных модулей, так как не полностью используются аппаратные

средства защиты процессора. В Win 95 аппаратно защищены только модули ядра, виртуальные драйвера, драйвера ФС и некоторые другие компоненты, расположенные между 3-м и 4-м Гб. В области между 2-м и 3-м Гб запускаются приложения Win16, они совместно выполняются в едином адресном пространстве, поэтому их работа так же ненадёжна, как и в Win 32.

Windows NT



Win NT намного устойчивее, так как использует аппаратные средства защиты памяти, в отличие от Win 9x.

Последний модуль (ядро, драйверы и т.п.) – отвечает за работу ФС, его код не виден для прикладных программ. Приложения Win 16 работают в сеансе windows on windows, каждое из приложений выполняется в собственном адресном пространстве, так есть возможность выполнения программ в одном адресном пространстве.

В Win NT существует виртуальная DOS-машина (VDM), используется для запуска программ, предназначенных для MS-DOS. В Win 95 возможен переход в однозадачный режим (MS-DOS), для этого потребуется перезагрузка системы. Возможен запуск нескольких сеансов VDM. Распределением памяти в NT занимается диспетчер виртуальной памяти (VMM), он старается уменьшить число обращений к HDD: поддерживается страничная организация памяти; обеспечивает замещение загрузочных страниц (обработка события «нет свободных страниц»). Когда процесс использует код или данные, находящиеся в ОП, диспетчер памяти резервирует место в файле подкачки, для того, что легко можно было выгрузить данные из памяти в этот файл. Рекомендация – размер файла подкачки (swap) = размер ОП + 12 Мб. В Win NT объекты хранятся в «пулах» памяти (memory pools). *Существуют 2 вида:*

1. Перемещаемый пул. Можно выгрузить в swap-файл;
2. Неперемещаемый пул. В нём содержатся структуры данных, используемые для обработки прерываний; системные структуры, используемые для работы в многопроцессорных конфигурациях (для предотвращения конфликтов).

Вся виртуальная память в NT делится на 3 класса:

1. *Зарезервированная память.* Набор адресов, которые VMM выделяет для процесса, но не учитывает их в квоте памяти процессов до тех пор, пока они не будут использованы реально занятым процессом. Если процессу необходима дополнительная память, то в случае наличия доступной памяти процессу выделяется больший её объём.
2. *Выделенная память.* Объём выделенной памяти процесса характеризуется фактически занятым процессом объёмом. Для выделения памяти резервируется место в swap-файл и ограничивается его размером.

Доступная память. Остальная часть памяти.

Самостоятельная работа №27
«Видеоадаптеры»

Задание 1. Оформление практических заданий в отчет

Самостоятельная работа №28
« Память»

1. Задание. Написать и отладить программу оконного приложения

Студент должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- выполнять отладку и тестирование программы на уровне модуля;

знать:

- основные этапы разработки программного обеспечения;
- основные принципы отладки и тестирования программных продуктов;

Критерии оценки:

- оценка «отлично», выставляется если программа работает, отчет выполненной работы оформлен полностью.
- оценка «хорошо», выставляется, если программа работает, отчет оформлен частично.
- оценка «удовлетворительно» выставляется, если программа работает, но имеются погрешности в коде или не все компоненты функционируют, отчет оформлен частично.
- оценка «неудовлетворительно» выставляется, если программа не работает, отчет не оформлен.

Пример выполнения

1. Задание. Написать и отладить программу оконного приложения

Написать программу обработки файла, содержащего информацию о рейтинге студентов. Каждая запись должна содержать Ф.И.О. и полученный балл рейтинга. Вывести информацию, отсортированную в порядке увеличения рейтинга. Результаты выполнения программы сохранить в текстовом файле. При работе с файлом должны быть выполнены следующие действия: создание, просмотр, добавление новой записи, сортировка, сохранение результатов.

Создание оконного приложения

Настройка компонент OpenDialog и SaveDialog

На странице *Dialogs* выбрать пиктограммы  ,  для установки компонент *OpenDialog* и *SaveDialog* соответственно.

Для выбора нужных файлов установить фильтры следующим образом: выбрав компоненту, дважды щелкнуть кнопкой мыши по правой части свойства *Filter* инспектора объектов, и в появившемся окне *Filter Editor*, в левой части записать текст, характеризующий выбор, в правой части – *маску*. Для *OpenDialog*1 установить значения *маски*, как показано на рис.1. Формат **.dat* означает, что будут видны все файлы с расширением *dat*, а формат **.** – будут видны все файлы (с любыми именами и расширениями).

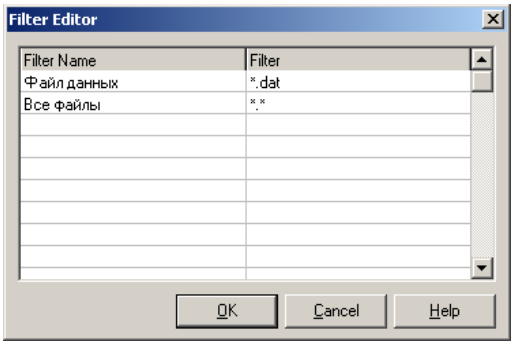


Рис. 1

Для того чтобы файл автоматически записывался с расширением *dat*, в свойстве *DefaultExt* записать требуемое расширение – *.dat*.

Аналогичным образом настраивается *SaveDialog*1 для текстового файла, который будет иметь расширение *.txt*.

Работа с программой

Форма может иметь вид, представленный на рис.2.

Кнопку «***Создать***» нажимаем только при первом запуске программы или, если захотим заменить прежнюю информацию на новую, в окне *Memo*1 отображается путь и имя созданного файла.

Заполнив оба поля информацией, нажимаем кнопку «***Добавить***», после чего введенная информация отображается в окне *Memo*1.

Для работы с уже созданным файлом нажимаем кнопку «***Открыть***» – в *Memo*1 выводится содержимое всего файла, после чего можно добавлять новые данные в конец этого файла, не уничтожая предыдущие.

При нажатии кнопки «***Сортировать***» в *Memo*1 выводятся записи, сортированные по возрастанию рейтинга.

При нажатии кнопки «***Сохранить результаты***» создается текстовый файл, в котором сохранится информация, выведенная в *Memo*1. Этот файл можно просмотреть в любом текстовым редакторе (блокноте, *Word*).

В текст программы включена пользовательская функция void ***Out***(*TZap, TMemo**); – для вывода в *Memo*1 одной записи.

Для создания результирующего текстового файла используется функция, ***SaveToFile***(*FileNameRez*); позволяющая записать все содержимое *Memo*1 в файл с указанным именем.

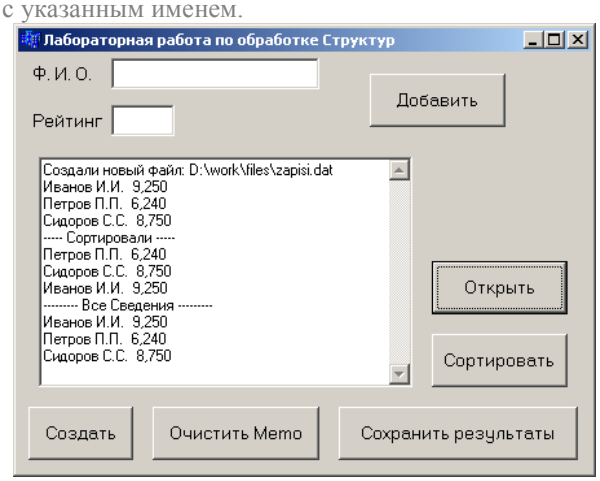


Рис.2

Текст программы может иметь следующий вид:

```
        . . .
#include <stdio.h>
#include <io.h>
        . . .
//---------------------------------------------------------------------------
struct TZap{
                        char FIO[30];
                        double s_b;
```

```
    } Zap;
int size = sizeof(TZap);
FILE *Fz;
AnsiString File_Zap;
void Out(TZap, TMemo*);
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
                        Edit1->Text="";       Edit2->Text="";
Memo1->Clear();
}
//------------------ Создать-------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
      OpenDialog1->Title="Создать новый файл";
      if (OpenDialog1->Execute()){
            File_Zap = OpenDialog1->FileName;
            if ((Fz=fopen(File_Zap.c_str(),"wb"))==NULL)  {
                  ShowMessage("Ошибка создания ФАЙЛА!");
                  return;
            }
      }
      Memo1->Lines->Add("Создали новый файл: "+AnsiString(File_Zap));
      fclose(Fz);
}
//----------------- Добавить-----------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
      Fz = fopen(File_Zap.c_str(),"ab");
      strcpy(Zap.FIO,  Edit1 -> Text.c_str());
      Zap.s_b = StrToFloat(Edit2->Text);
      Out(Zap, Memo1);
      fwrite(&Zap, size, 1, Fz);
      Edit1->Text=""; Edit2->Text="";
      fclose(Fz);
}
//--------------------- Сортировать ---------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
      TZap st, *mas_Z;
      Fz = fopen(File_Zap.c_str(),"rb");
      int D_f = fileno(Fz);                                 // Находим дескриптор файла
      int len = filelength(D_f);                            // Находим размер файла
      int i, j, kol;
      kol = len/size;                                       //Количество записей в файле
      mas_Z = new TZap[kol];
// Считываем записи из файла в динамический массив
      for (i=0; i < kol; i++)
                        fread((mas_Z+i), size, 1, Fz);
      fclose(Fz);
                              Memo1->Lines->Add("Сортированные сведения");
      for (i=0; i < kol-1; i++)
            for (j=i+1; j < kol; j++)
                  if (mas_Z[i].s_b > mas_Z[j].s_b) {
                                    st = mas_Z[i];
                                    mas_Z[i] = mas_Z[j];
                                    mas_Z[j] = st;
                  }
              for (i=0; i<kol; i++)
                        Out(mas_Z[i], Memo1);
   delete []mas_Z;
}
//------------------ Сохранить ----------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
```

```
{
        SaveDialog1->Title="Сохранить файл результатов";
        if (SaveDialog1->Execute()) {
                AnsiString FileNameRez = SaveDialog1->FileName;
                Memo1->Lines->SaveToFile(FileNameRez);
        }
}
//-------------------- Открыть ----------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
  OpenDialog1->Title="Открыть файл";
  if (OpenDialog1->Execute()) {
              File_Zap = OpenDialog1->FileName;
              if ((Fz=fopen(File_Zap.c_str(),"rb"))==NULL)  {
                    ShowMessage("Ошибка открытия ФАЙЛА!");
                    return;
              }
  }
  Memo1->Lines->Add("--------- Все сведения ---------");
  while(1){
   if(!fread(&Zap,size,1,Fz)) break;
               Out(Zap, Memo1);
  }
  fclose(Fz);
}
//----------------- Очистка Memo -----------------------------------------
void __fastcall TForm1::Button6Click(TObject *Sender)
{
      Memo1->Clear();
}
//----------------- Функция вывода одной записи ----------------------
void Out(TZap z, TMemo *Memo1)
{
    Memo1->Lines->Add(AnsiString(z.FIO)+ "  "+FloatToStrF(z.s_b, ffFixed,6,3));
}
```

Создание консольного приложения

Для создания текстового файла в консольном приложении используем функцию *fprintf*(). Текст программы может иметь следующий вид:

```
. . .
#include <stdio.h>
#include <io.h>
. . .
struct TZap{
    char FIO[30];
    double s_b;
    } Zap;
int size = sizeof(TZap);
FILE *Fz, *Ft;
char File_Zap[] = "zapisi.dat";
char File_Rez[] = "rezult.txt";
void Out(TZap);

void main()
{
  int kod, D_f, i=0, j, kol;
  long len;
  TZap st, *mas_Z;
Ft = fopen(File_Rez, "w");
  while(true) {
              puts("\n Create – 1\n Add – 2\n View – 3\n Sort – 4\n EXIT – 0");
              scanf("%d", &kod);
              switch(kod) {
              case 1:
                        if ((Fz=fopen(File_Zap,"wb"))==NULL)  {
```

```
                        puts("\n Create ERROR!");
                        return;
                }
                fclose(Fz);
                printf("\n Create New File %s !\n",File_Zap);
        break;
        case 2:
                Fz = fopen(File_Zap,"ab");
                printf("\n F.I.O. – ");
                fflush(stdin);
                        gets(Zap.FIO);
                printf(" Ball – ");
                        scanf("%lf", &Zap.s_b);
                fwrite(&Zap, size, 1, Fz);
                fclose(Fz);
        break;
        case 3:
                if ((Fz=fopen(File_Zap,"rb"))==NULL)  {
                        puts("\n Open ERROR!");
                        return;
                }
// Вывод на экран
                printf("\n\t--------- Informations ---------");
// Запись такой же информации в текстовый файл Ft
                fprintf(Ft,"\n\t--------- Informations ---------");
                while(1) {
                        if(!fread(&Zap,size,1,Fz)) break;
                                Out(Zap);
                }
                fclose(Fz);
        break;
        case 4:
                Fz = fopen(File_Zap,"rb");
                D_f = fileno(Fz);
                        len = filelength(D_f);
                        kol = len/size;
                        mas_Z = new TZap[kol];
// Считываем записи из файла в динамический массив
                        for (i=0; i < kol; i++)
                        fread((mas_Z+i), size, 1, Fz);
                        fclose(Fz);
                        printf("\n\t----- S O R T -----\n");
                        fprintf(Ft,"\n\t----- S O R T -----\n");
                        for (i=0; i < kol-1; i++)
                                for (j=i+1; j < kol; j++)
                                        if (mas_Z[i].s_b > mas_Z[j].s_b) {
                                                st = mas_Z[i];
                                                mas_Z[i] = mas_Z[j];
                                                mas_Z[j] = st;
                                        }
                                        for (i=0; i<kol; i++)
                                        Out(mas_Z[i]);
                        delete []mas_Z;
                break;
        case 0:
                fclose(Ft);
                return;
        }
  }
}
//----------------- Функция вывода одной записи на экран и в файл --------------------
void Out(TZap z)
{
    printf("\n %20s , %6.3lf .", z.FIO,z.s_b);
```

```
    fprintf(Ft, "\n %20s , %6.3lf .", z.FIO, z.s_b);
}
```

Первоначально выбав пункт «1», создаем файл с именем *zapisi.dat*, который будет располагаться в текущем каталоге (созданной папке). Затем, выбирая пункт «2», последовательно вводим 4 записи. Выбрав пункт «3», просматриваем содержимое файла, а сортированные записи выведем на экран (запишем в файл), выбрав пункт «4». Результаты выполнения программы могут иметь вид:

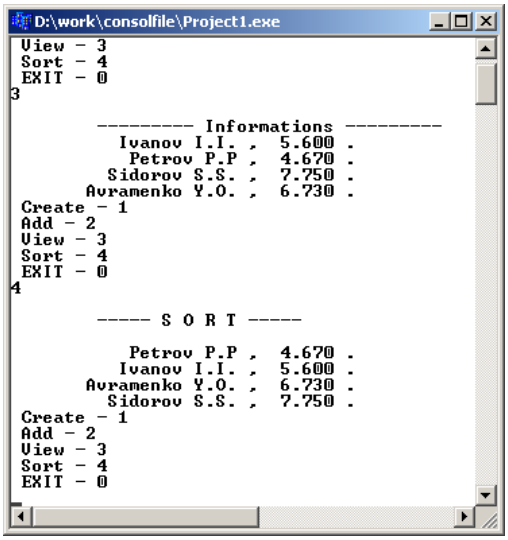


Рис.3

Индивидуальные задачи к заданию 1.

Написать программу обработки файла типа запись, содержащую следующие пункты меню: «Создание», «Просмотр», «Коррекция» (добавление новых данных или редактирование старых), «Решение индивидуального задания».

Каждая запись должна содержать следующую информацию о студентах:

– фамилия и инициалы;

– год рождения;

– номер группы;

– оценки за семестр: по физике, математике, информатике, химии;

– средний балл.

Организовать ввод исходных данных, средний балл рассчитать по введенным оценкам. Содержимое всего файла и результаты решения индивидувльного задания записать в текстовый файл.

1. Распечатать анкетные данные студентов, сдавших сессию на 8, 9 и 10.
2. Распечатать анкетные данные студентов-отличников, фамилии которых начинаются с интересующей вас буквы.
3. Распечатать анкетные данные студентов-отличников из интересующей вас группы.
4. Распечатать анкетные данные студентов, фамилии которых начинаются с буквы *А*, и сдавших математику на 8 или 9.
5. Распечатать анкетные данные студентов, имеющих оценки 4 или 5 по физике и оценку больше 8 по остальным предметам.
6. Распечатать анкетные данные студентов интересующей вас группы. Фамилии студентов начинаются с букв *В*, *Г* и *Д*.
7. Распечатать анкетные данные студентов, не имеющих оценок меньше 4 по информатике и математике.
8. Вычислить общий средний балл всех студентов и распечатать список студентов со средним баллом выше общего среднего балла.
9. Вычислить общий средний балл всех студентов и распечатать список студентов интересующей вас группы, имеющих средний балл выше общего среднего балла.
10. Распечатать анкетные данные студентов интересующей вас группы, имеющих неудовлетворительную оценку (меньше 4).
11. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 9 или 10 по информатике.
12. Распечатать анкетные данные студентов, имеющих оценки 7 или 8 по физике и оценки 9 или 10 по высшей математике.
13. Вычислить общий средний балл студентов интересующей вас группы и распечатать список студентов этой группы, имеющих средний балл выше общего.
14. Распечатать анкетные данные студентов-отличников интересующей вас группы.
15. Распечатать анкетные данные студентов интересующей вас группы, имеющих средний балл выше введенного с клавиатуры.
16. Распечатать анкетные данные студентов интересующей вас группы, имеющих оценку 8 по физике и оценку 9 по высшей математике.

Самостоятельная работа №29

. «Программный сегмент и программный идентификатор»

1. Задание. Составить и отладить программу обработки списка записей.

Студент должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- выполнять отладку и тестирование программы на уровне модуля;

знать:

- основные этапы разработки программного обеспечения;
- основные принципы отладки и тестирования программных продуктов;

Критерии оценки:

- оценка «отлично», выставляется если программа работает, отчет выполненной работы оформлен полностью.
- оценка «хорошо», выставляется, если программа работает, отчет оформлен частично.
- оценка «удовлетворительно» выставляется, если программа работает, но имеются погрешности в коде или не все компоненты функционируют, отчет оформлен частично.
- оценка «неудовлетворительно» выставляется, если программа не работает, отчет не оформлен.

Пример выполнения:

1. Задание. Составить и отладить программу обработки списка записей.

Пусть требуется разработать программу для обработки списка записей, представляющих собой структуру данных, в которой добавление элементов производится в конец списка (как в очереди), а удаление элементов – по заданному ключу (значению информационного поля). Для ввода и отображения значений информационных полей списка в форме разместим следующие визуальные объекты Borland C++ Builder: *Edit1* - *Edit3* – для задания значений информационному полю соответственно первого, добавляемого и заданного элемента; *Memo1* – для вывода и отображения списка (значений информационных полей элементов списка), *Label1* - *Label5* – для ввода и отображения пояснений назначения выбранных объектов. Для управления работой программы в форме разместим объекты *Button1* - *Button3* – для задания действий, соответствующих операциям по обработке списка: создание (инициализация) списка из одного элемента, добавление нового элемента в конец списка, удаление заданного элемента списка; *Button4* – для выхода из программы.

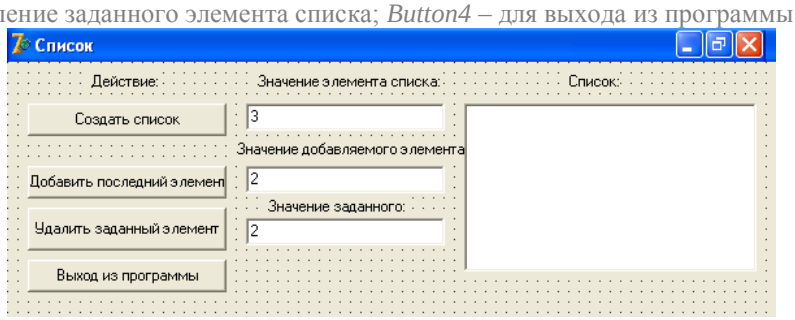


Рис.1. Окно формы с выбранными объектами

Для того чтобы программа в ответ на нажатие кнопки выполняла именно то действие, которое требуется, в окне кода вносим соответствующий программный код для обработчика события *Click* для каждой кнопки.

В результате получаем следующий исходный код программы для обработки списка записей.

Файл *Unit.h*:

```
/* Программа обработки списка записей */
#ifndef Unit1H
        #define Unit1H
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
class TForm1 : public TForm
{
__published: // IDE-managed Components
TButton *Button1;
TButton *Button2;
TButton *Button3;
TButton *Button4;
TLabel *Label1;
TLabel *Label2;
TEdit *Edit1;
TEdit *Edit2;
TEdit *Edit3;
TLabel *Label3;
TLabel *Label4;
TMemo *Memo1;
```

```
TLabel *Label5;
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall Button3Click(TObject *Sender);
void __fastcall Button4Click(TObject *Sender);
private: // User declarations
public: // User declarations
__fastcall TForm1(TComponent* Owner);
};
extern PACKAGE TForm1 *Form1;
            #endif
```

Файл *Unit.cpp*:

```
#include <vcl.h>
#pragma hdrstop
#include "SpisokOper.h"
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
bool made;
Node *Ph = NULL;
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{ /* Метод для инициализации списка из одного элемента */
Memo1->Lines->Clear();
Ph = init(Ph, &made, Edit1);
outp(Ph, 1, Memo1);
}
void __fastcall TForm1::Button2Click(TObject *Sender)
{ /* Метод для идобавления элемента в конец списка */
if(made) {
Memo1->Lines->Clear();
addLast(Ph,Edit2);
Memo1->Lines->Clear();
outp(Ph, 1, Memo1);
            }
else {
ShowMessage("Сначала создайте список!");
return;
}
}
void __fastcall TForm1::Button3Click(TObject *Sender)
{ /* Метод для удаления заданного элемента из списка */
if(Edit3->Text.IsEmpty()) {
ShowMessage("Неверный ввод! Повторите.");
return;
}
Ph = delet(Ph, Edit3->Text);
Memo1->Lines->Clear();
outp(Ph, 1, Memo1);
}
void __fastcall TForm1::Button4Click(TObject *Sender)
{ /* Метод для удаления списка (освобождение паямяти) */
disp(Ph);
Close();
}
```

Файл *SpisokOper.h*:

```
/
```

Модуль с процедурами обработки списка:

Инициализации списка
Добавление элемента в конец списка
Удаление заданного элемента
Отображение (печать) списка в окне
Удаление списка (освобождение памяти)
/

```
#ifndef SpisokOperH
        #define SpisokOperH
struct Node {
AnsiString inf;
Node *next;
};
Node *init(Node *p, bool *pmade, TEdit *Edt);
void outp(Node *p, int t, TMemo *mem);
void addLast(Node *p, TEdit *Edt);
Node *delet(Node *p, AnsiString a);
void disp(Node *p);
#endif
```

Файл *SpisokOper.cpp*:

```
#include <vcl.h>
#pragma hdrstop
#include "SpisokOper.h"
void outp(Node *p, int t, TMemo *mem)
{ /
Вывод списка в окно Mem
p - указатель на первый элемент списка
/
Node *q;
q = p;
if(q!=NULL) {
mem->Lines->Add(q->inf);
outp(q->next, t+1, mem);
}
}
Node *init(Node *p, bool *pmade, TEdit *Edt)
{ /
Инициализация списка из одного элемента
        p - указатель на первый элемент списка
/
if(*pmade) {
ShowMessage("Список был уже создан ранее!");
return p;
}
if(Edt->Text.IsEmpty()) {
ShowMessage("Неверный ввод! Повторите.");
return NULL;
}
p = new(Node);
p->inf = Edt->Text;
p->next = NULL;
*pmade = true;
return p;
}
void addLast(Node *p, TEdit *Edt)
{ /
добавление элемента в конец списка
p - указатель на некоторый элемент списка
/
Node *q, *r;
r = p->next;
if(r==NULL) {
if(Edt->Text.IsEmpty()) {
ShowMessage("Неверный ввод! Повторите.");
return;
```

```
}
q = new(Node);
q->inf = Edt->Text;
p->next = q;
q->next = NULL;
        }
else addLast(r, Edt);
}
Node *delet(Node *p, AnsiString a)
{ /
поиск и удаление заданного элемента из списка
p - указатель на первый элемент списка
/
Node *q, *q0;
q = p;
while(q!=NULL)
{ if(q->inf==a)
{ ShowMessage("Удален элемент "' + q->inf + "' Нажми ОК");
if(p==q) p = q->next; else q0->next = q->next;
delete q;
return p;
}
q0 = q;
q = q->next;
}
}
void disp(Node *p)
{ /
освобождение памяти из под списка
p - указатель на первый элемент списка
/
Node *q, *r;
q = p;
if(q!=NULL) {
r = q->next;
delete q;
disp(r);
}
        p = NULL;
}
```

В модуле *SpisokOper* приведенной программы в виде процедур реализованы следующие операции: создание списка из одного элемента (функция *init*), добавление элемента в конец списка (функция *addLast*), поиск и удаление найденного элемента из списка (функция *delet*), вывод значений элементов списка на экран монитора (функция *outp*), удаление списка и освобождение памяти (функция *disp*). При запуске программы на выполнение открывается окно, показанное на рис.2. Прежде чем начать работу со списком, его следует создать. Для этого в поле объекта *Edit1* вводится значение информационного поля первого элемента и нажимается кнопка *Button1*. В результате в динамической памяти будет создан список из одного элемента, значения информационного поля которого будет отображено в объекте *Memo1*. При добавлении элемента в список следует сначала ввести значение его информационного поля в объект *Edit2*, после чего нажать кнопку *Button2*. Новый элемент будет добавлен в конец списка, а в поле объекта *Memo1* отобразится значения информационных полей имеющихся элементов списка. Для удаления элемента сначала в объекте *Edit3* указывается значение заданного элемента, после чего нажимается кнопка *Button3*. В результате будет удален первый из элементов списка, значение информационного поля которого совпадает с заданным, а в объекте *Memo1* будут отображены значения информационных полей всех оставшихся элементов списка.

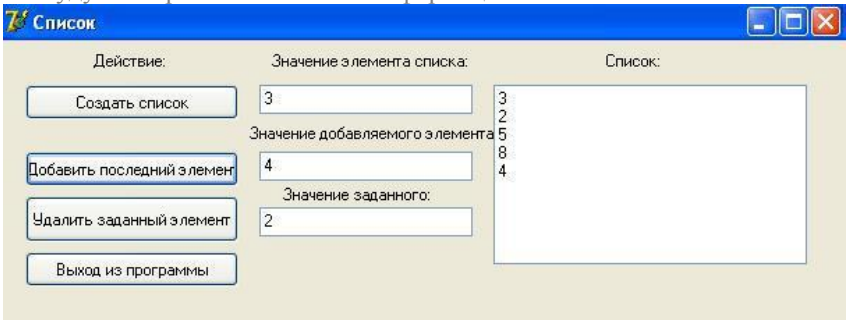


Рис. 2. Окно программы для обработки списка записей

Индивидуальные задачи к заданию 1.

1) Стек: создание, добавление, исключение элементов и функция, определяющая не пуст ли стек (добавление в начало, удаление первого элемента списка).

2) Очередь: создание, добавление, исключение элементов и функция, определяющая не пуста ли очередь (добавление в конец списка, удаление первого элемента).

3) Список: создание, удаление первого элемента, добавление после заданного элемента.

4) Список: создание, удаление последнего элемента, добавление перед заданным.

5) Список: создание, добавление первого элемента, удаление после заданного.

6) Список: создание, добавление последнего элемента, удаление перед заданным.

7) Список: создание, добавление элементов в конец списка, поиск по значению информационного поля максимального и минимального элементов и их перестановка.

8) Двусвязный список: создание, добавление в начало, удаление первого элемента списка.

9) Двусвязный список: создание, добавление в конец списка, удаление первого элемента.

10) Двусвязный список: создание, удаление первого элемента, добавление после заданного.

11) Двусвязный список: создание, удаление последнего элемента, добавление пе-ред заданным.

12) Двусвязный список: создание, добавление в начало списка, удаление после заданного.

13) Двусвязный список: создание, добавление в конец списка, удаление элемента перед заданным.

14) Двусвязный список: создание, добавление элементов в конец списка, поиск по значению информационного поля максимального и минимального элементов и их перестановка.

15) Двусвязный список: создание и упорядочение элементов.

16) Упорядоченный по значению информационного поля список: создание, добавление, элементов с сохранением упорядоченности.

Самостоятельная работа №30
. «Системные управляющие блоки»

1.Задание: Подготовить информационное сообщение на тему:

«Распределение памяти DOS»

«Управление программами»

«Дисковая структура DOS»

Студент должен:

- **знать,** драйверы устройств буферизации дискового ввода-вывода, связи системного блока, префикс программного сегмента и его структура, назначение его полей.

- **уметь,** работать с литературой и выявлять главную часть, анализируя полученную информацию.

Критерии оценки подготовки сообщения:

- полнота и качественность информации по заданной теме;
- свободное владение материалом сообщения;
- логичность и четкость изложения материала;
- использование фактов при изложении материала, примеров, жизненных ситуаций;
- наличие и качество презентационного материала.

Пример выполнения

1. Задания на тему: «Распределение памяти DOS»

Один из самых сложных процессов в операционной системе – обслуживание памяти. Это связано с тем, что MS-DOS должна иметь четкое представление о том, какая часть памяти находится в работе, а какая - доступна для использования. Сушествует три фундаментальных требования, которым должна отвечать операционная система при работе с памятью:

1. Выделять свободные блоки (распределять память) для работающих программ.
2. При необходимости изменять размер ранее распределенных блоков.
3. Освобождать используемые блоки при завершении выполнения занимающих их программ (перераспределять память).

Чтобы удовлетворить перечисленным требованиям, в MS-DOS имеется группа специальных функций. Это функции 48H (распределение памяти), 49H (очистка памяти) и 4AH (перераспределение памяти). Вспомним материал из предыдущего раздела. Для изменения размера операционной среды файла COMMAND.COM использовалась функция 48H. С ее помощью для операционной среды файла выделялся свободный блок памяти.

Первый параграф каждого выделяемого блока отводится для блока управления памятью (mcb). В первый байт блока записывается либо значение 4DH, либо 5AH. Если первый байт блока равен 4DH, то mcb является внутренним членом цепочки, связывающей mcb всех задействованных блоков.

Если он равен 5AH, то данное mcb является последним в цепочке.

Второй и третий байты mcb отводятся под идентификатор процесса (PID), занимающего данный блок (значение идентификатора записывается в "обратном порядке"). Напомним, что PID - это адрес сегмента psp (или адрес сегмента программы).

В четветый и пятый байты mcb записывается количество параграфов в данном блоке памяти (значение записывается в "обратном порядке"). Складывая это значение с адресом данного mcb, получаем адрес следующего mcb в цепочке.

Как отмечалось выше, доступ к mcb в MS-DOS осуществляется с помощью трех системных функций. Фирмы IBM и Microsoft не рекомендуют работать с mcb впрямую. Благодаря стараниям этих фирм, такая работа сильно затруднена.

Например, прикладная программа не может обрабатывать mcb. Однако, любой программист может просмотреть содержимое нужного ему mcb и воспользоваться хранящейся в нем информацией.

К несчастью, не существует официально рассмотренного способа доступа к блокам mcb. Не имеется даже только что приведенного описания блока. Однако, программисты нашли способ доступа к блокам управления памятью - с помощью системной функции 52H. Эта функция возвращает указатель на первый mcb в цепочке распределенных блоков памяти. И если найдено первое звено, то можно проследить всю цепочку.

Рассмотрим, как можно использовать информацию в блоках mcb. Для этого войдем в дебаггер. Стартуйте DEBUG (команда debug) и ждите появления запроса (-). С его появлением введите команду "assembler" (или "a 100"). На экране появится примерно следующее:

-a 100 1259:0100

Критерий оценки:

правильно выполнены 2 задания – оценка «5» выполнены 2 задания, одно с ошибками – оценка «4»правильно выполнено 1 задание – оценка «3»