Ф.4.02-02

Министерство образования и науки Республики Казахстан
Кокшетауский государственный университет им. Ш. Уалиханова
Кафедра Информационных систем и вычислительной техники

Баклхазова У.У

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам по дисциплине

Технология программирования

для студентов специальности
5В070300 – Информационные системы,
5В070400 – Вычислительная техника и
программное обеспечение

Форма обучения: **очная, заочная**

Кокшетау 2014 г.

Баклхазова У.У

Методические указания к лабораторным работам по дисциплине

«Технология программирования»

Кокшетау: КГУ им. Ш. Уалиханова, 2014-2015.

Методические указания составлены в соответствии с требованиями учебного плана и программой дисциплины «Технология программирования» и включают все необходимые сведения по выполнению тем лабораторных работ курса.

Методические указания предназначены для студентов специальностей 5В070300 –«Информационные системы», 5В070400- «Вычислительная техника и программное обеспечение»

Утверждено на заседании кафедры
«Информационных систем и вычислительной техники»

«_» ____ ____/ Протокол № /

Заведующий кафедрой _____ Хан С. И.

Утверждено учебно-методическим советом
Факультета техники и технологии

«____» _____ 2014-2015 г. / Протокол № /

Председатель УМК_____ Булашева А.И.

Содержание

Введение

Учебно-методические указания предназначены для изучения данной дисциплины в соответствии с государственным стандартом специальности.

Целью преподавания дисциплины "Технология программирования" является получение студентами знаний по организации основных этапов решения задач на ЭВМ, способам конструирования программ с применением языков высокого уровня и основам доказательства их правильности.

В результате изучения дисциплины студенты получат знания об основных этапах решения задачи на ЭВМ, о критериях качества программы, о спецификациях программ, будут знать основные методы и средства разработки алгоритмов и программ, приемы структурного программирования, способы записи алгоритма на языке высокого уровня. Студенты приобретут навыки проектирования архитектуры и разработки функциональных модулей, пакетов программ, разработки программной документации в соответствии со стандартами, получат опыт выбора технологии и инструментальных средств. Будущие специалисты должны научиться разрабатывать программы на языках высокого уровня для задач обработки числовой и символьной информации, а также должны проводить отладку и тестирование разработанных программ.

Общие указания к выполнению лабораторных работ.

Основная цель, которая ставиться студенту при выполнении задания - практическое освоение всех этапов разработки надежной программы для решения задачи на ПЭВМ, начиная от анализа условия задачи и заканчивая сдачей отчета по написанной программе. Каждая лабораторная работа состоит из одной или 2-х задач и включает следующие виды работ:

1. Анализ условия задачи и выработка подхода к ее решению. Обоснование алгоритма.
2. Пошаговая разработка алгоритма решения и его описание.
3. Составление блок-схемы алгоритма.
4. Выбор и обоснование представления для входных, выходных и промежуточных данных.
5. Кодирование алгоритма (запись на языке Pascal, и на языке C++).
6. Выбор набора тестов, на которых будет проверяться программа.
7. Отладка программы и демонстрация правильной ее работы на выбранном наборе тестов.

Обратите внимание на то, что для повышения эффективности составления алгоритма относительно больших программ применяется структурный подход к программированию. Это способствует уменьшению затрат на создание и дальнейшее использование программ при эксплуатации. Структурный подход к программированию состоит из трех частей: нисходящая разработка, структурное программирование и сквозной контроль (тестирование).

При нисходящей разработке проектирование предусматривает сначала определение задачи в общих чертах, а затем задача разбивается на ряд более простых подзадач. Для каждой подзадачи составляется алгоритм ее решения.

В структурном программировании заданы правила соединения типовых конструкций. Структура - это оператор (вполне определенный, не всякий) языка программирования, который имеет один вход и один выход.

Подготовка к каждой лабораторной работе производится во внеаудиторное время. Выполнив лабораторную работу, студент оформляет отчет, который состоит из следующих разделов:

1. Тема и цель работы.
2. Условия задания.
3. Схема алгоритма решения задачи: математическая модель задачи, блок схема алгоритма.
4. Анализ алгоритма.
5. Текст программы и размещение исходных данных при вводе.
6. Результаты выполнения программы.
7. Обоснование правильности разработанной программы.
8. Выводы.

При защите отчета необходимо отвечать на контрольные вопросы и уметь пояснять работу программы.

Правила техники безопасности при выполнении лабораторных работ

1. Общие требования безопасности.

Опасные производственные факторы: воздействие на человека электрического тока, электрического поля, рентгеновского излучения, ультрафиолетового излучения.

Действие факторов: вследствие неисправности кабеля, замыкания в цепи пользователь компьютера попадает под напряжение.

2. Требования безопасности перед началом работы.

2.1. К работе на компьютере допускаются только лица, прошедшие инст-руктаж по правилам пользования компьютера.

2.2. Следует убедиться в отсутствии видимых повреждений аппаратуры и рабочего места.

3. Требования безопасности во время работы.

3.1. Необходимо соблюдать оптимальное расстояние от глаз до экрана монитора (60-70 см.), допустимо не менее 50 см.

3.2. При работе на компьютере следует сидеть прямо, с небольшим наклоном вперед, не сутулясь. Величина угла в суставах не должна быть менее 90°.

3.3. При внезапном отключении электроэнергии в сети выключить ком-пьютер.

3.4. Во время эксплуатации при повреждении штепсельного соединения, токопроводящего кабеля, появлении дыма (огня) из компьютера, обнаружения замыкания на корпус следует немедленно отключить компьютер и доложить о поломке преподавателю или дежурному лаборанту.

3.5. Оптимальное время непрерывной работы на компьютере 40-50 мин. По истечении этого времени необходимо сделать перерыв на 10 минут.

3.6. Строго запрещается: трогать разъёмы соединительных кабелей, прикасаться к экрану и к тыльной стороне монитора, включать и выключать аппаратуру без указания преподавателя,

3.7. Строго запрещается класть книги, тетради на монитор и клавиатуру, работать во влажной одежде и влажными руками.

4. Требования безопасности при аварийной обстановке.

4.1. Необходимо выключить компьютер.

4.2. Нельзя пользоваться компьютером до полного устранения неисправности.

4.3. При получении травмы и внезапном заболевании следует немедленно известить преподавателя или лаборанта.

5. Требования безопасности по окончании работы.

5.1. Необходимо выключить компьютер.

5.2. Обо всех замечаниях и недостатках в работе компьютера следует сообщить преподавателю или дежурному лаборанту.

Лабораторная работа №1
Программирование линейных алгоритмов

Цель работы: закрепление знаний по написанию программ на Pascal, выработать практические навыки работы на языке C++, научиться создавать, вводить в компьютер, выполнять и исправлять простейшие программы на ЯП в режиме диалога, познакомиться с диагностическими сообщениями компилятора об ошибках при выполнении программ, реализующих линейные алгоритмы.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения:

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное число действий. Таким образом, некоторые инструкции должны выполняться строго после завершения работы инструкций, от которых они зависят. Независимые инструкции или инструкции, ставшие независимыми из-за завершения работы инструкций, от которых они зависят, могут выполняться в произвольном порядке, параллельно или одновременно, если это позволяют используемые процессор и операционная система.

В любом алгоритме для обозначения данных используют некоторый набор символов, называемых *буквами*. Конечную совокупность букв называют *алфавитом*, из любой конечной последовательности которого можно составить *слово*, т.е. в любом алфавите реальным данным можно сопоставить некоторые слова, в дальнейшем обозначающие эти данные.

При словесной записи алгоритм описывается с помощью естественного языка с использованием следующих конструкций:

1) шаг (этап) обработки (вычисления) значений данных – «=»;

2) проверка логического условия: если (условие) истинно, то выполнить действие 1, иначе – действие 2;

3) переход (передача управления) к определенному шагу (этапу) N.

Линейным называется алгоритм, в котором результат получается путем однократного выполнения заданной последовательности действий при любых значениях исходных данных. Операторы программы выполняются последовательно, один за другим, в соответствии с их расположением в программе.

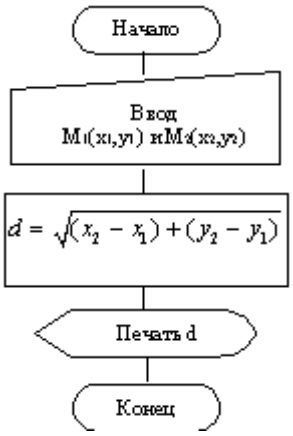
Пример1: Определить расстояние на плоскости между двумя точками с заданными координатами M1(x1,y1) и M2(x2,y2)

Этапы решения задачи:

1. Математическая модель: расстояние на плоскости между двумя точками M1(x1,y1) и M2(x2,y2) высчитывается по формуле $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

2. Составим схему алгоритма

Уточним содержимое блока "Вычисление и печать расстояния"



3. Переводим блок-схему на язык Паскаль.

```
program example1;
var x1, x2, y1, y2: integer;
   d:real;
begin
    write('x1= '); readln(x1);
    write('y1= '); readln(y1);
    write('x2= '); readln(x2);
    write('y2= '); readln(y2);
    d:=sqrt(sqr(x2-x1)+sqr(y2-y1));
writeln('d=',d);
end.
```

Доработаем программу, так чтобы она обладала некоторым интерфейсом:

```
program example1;
var x1, x2, y1, y2: integer; d:real;
begin
writeln('Эта программа вычисляет расстояние между двумя точками на плоскости');
writeln('Введите координаты двух точек:');
write('x1= '); readln(x1);
write('y1= '); readln(y1);
write('x2= '); readln(x2);
write('y2 ='); readln(y2);
d:=sqrt(sqr(x2-x1)+sqr(y2-y1));
writeln('d= ',d);
writeln('нажмите Enter для завершения работы программы');
readln;
end.
```

Программа, написанная на языке С++, состоит из одной или нескольких функций, одна из которых имеет идентификатор ***main**** – главная (основная). Она является первой выполняемой функцией (с нее начинается выполнение программы) и ее назначение – управлять работой всей программы (проекта).

Общая структура программы на языке С++ имеет вид:

<директивы препроцессора>
<определение типов пользователя – *typedef*>
<описание прототипов функций>
<определение глобальных переменных>
<функции>

В свою очередь, каждая функция имеет следующую структуру:
<класс памяти> <тип> < *ID* функции> (<объявление параметров>)

{ – начало функции
код функции
} – конец функции

Переводим блок-схему на язык С++.

```
#include <iostream>
#include <string>
#include <math.h>
using namespace std;
int main(int argc, char* argv[])
{
    double x1,x2,y1,y2,distance;
    cout <<"Введите координату x1:"<<endl;
    cin>>x1;
    cout <<"Введите координату y1:"<<endl;
    cin>>y1;
```

```
        cout <<"Введите координату x2:"<<endl;
        cin>>x2;
        cout <<"Введите координату y2:"<<endl;
        cin>>y2;  distance=sqrt(pow((x2-x1),2)+pow((y2-y1),2));
  // формула для вычисления расстояния
          cout<<"Расстояние между двумя точками:"<<distance<<endl;
  }
```

Контрольные вопросы:

1. Каковы назначение и возможности ЯП Pascal, C++?
2. Как запустить программу на трансляцию и выполнение?
3. Как записываются операторы начала и конца программы?
4. Из каких разделов состоит программа на языке Pascal, C++?
5. В какой последовательности должны быть записаны разделы программы на языке Pascal, C++?
6. Как записываются операторы вывода на экран в Pascal, C++?

Задания по теме «Составление линейных алгоритмов»

Составить программу для расчета двух значений z_1 и z_2. Ввод исходных данных можно задавать при декларации или вводить с клавиатуры. Игнорировать возможность деления на ноль. Значение π = 3,1415926.

Вариант 1	$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha),\quad z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$
Вариант 2	$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha\ ,\quad z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right)$
Вариант 3	$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha},\quad z_2 = 2\sin\alpha$
Вариант 4	$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\beta}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\beta}{4}\right),\quad z_2 = \frac{\sqrt{2}}{2}\sin\frac{\beta}{2}$
Вариант 5	$z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha\ ,\quad z_2 = \cos^2\alpha + \cos^4\alpha$
Вариант 6	$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha\ ,\quad z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \cdot \cos 4\alpha$
Вариант 7	$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right),\quad z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$
Вариант 8	$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1\ ,\quad z_2 = \sin(y + x) \cdot \sin(y - x)$
Вариант 9	$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2\ ,\quad z_2 = -4\sin^2\frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta)$

Вариант 10	$z_1 = \dfrac{\sin\left(\dfrac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$, $z_2 = ctg\left(\dfrac{5}{4}\pi + \dfrac{3}{2}\alpha\right)$

Задачи повышенной сложности

1. Студент начал решать задачи данного урока программирования, когда электронные часы показывали h1 часов и min1 минут, а закончил, когда было h2 часов и min2 минут. Составьте программу, позволяющую определить, сколько времени студент решал эти задачи. (Будем считать, что задачи решались не дольше суток.)

2. Найти максимум и минимум двух натуральных чисел,не используя ветвления алгоритма.

Литература основная

1. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
2. Малыхина М. П. Pascal 7.0:Практическое программирование. Решение типовых задач: Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
3. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си/ Е. М. Демидович. – Минск : Бестпринт, 2001

Лабораторная работа №2
Программирование разветвляющихся алгоритмов.

Цель работы: научиться правильно использовать условный оператор if; научиться составлять программы решения задач на разветвляющиеся алгоритмы.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Разветвляющийся алгоритм – это алгоритм, в котором выполнение того или иного действия (шага) зависит от выполнения или не выполнения какого-либо условия Достаточно часто то или иное действие должно быть выполнено в зависимости от значения логического выражения, выступающего в качестве условия.

В таких алгоритмах делается выбор: выполнять или не выполнять какую-нибудь группу команд в зависимости от условия, т.е. выбирается один из нескольких возможных путей (вариантов) вычислительного процесса. Каждый подобный путь называется ветвью алгоритма.

Разветвляющийся алгоритм может содержать несколько ветвей, отличающихся друг от друга содержанием вычислений. Выход

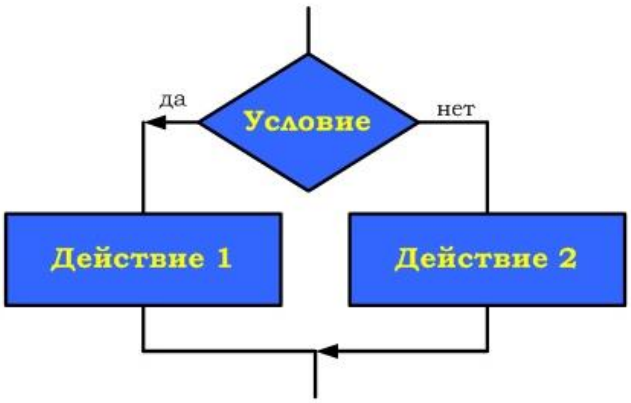
вычислительного процесса на ту или иную ветвь алгоритма определяется исходными данными задачи.

Признаком разветвляющегося алгоритма является наличие операций условного перехода, когда происходит проверка истинности некоторого логического выражения (проверяемое условие) и в зависимости от истинности или ложности проверяемого условия для выполнения выбирается та или иная ветвь алгоритма.

В логических выражениях используется операция сравнения: < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), = (равно), <> (не равно). Часто встречаются задачи, в которых используются не отдельные условия, а совокупность связанных между собой условий (отношений). Для связи используются AND и (или) OR.

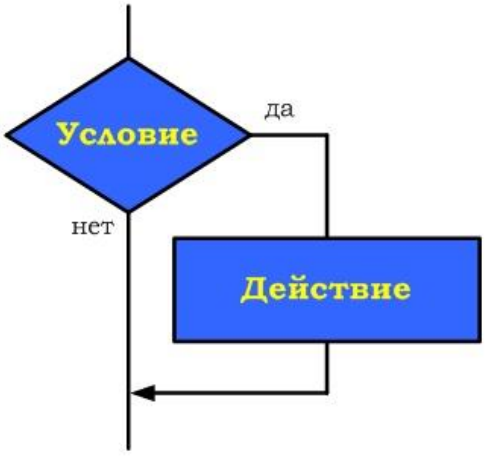
Алгоритм предполагает выполнение Действия 1, если записанное условие истинно (выполняется), и выполнение Действия 2, если условие ложно (не выполняется) – это полная развилка.

Полная развилка: If Условие then Действие 1 else Действие 2



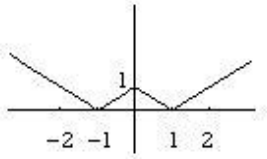
Если в алгоритме отсутствует Действие 2, т.е. если записанное условие истинно, то выполняется Действие 1, а если условие ложно, то никаких действий не выполняется – это не полная развилка.

Неполная развилка: If Условие then Действие 1



Перед выполнением работы необходимо ознакомится с правилами записи логических выражений, операций сравнения, операторов IF, CASE, SWITCH.

Пример1: Дано действительное a. Для функций f(a), график которой представлен на рисунке, вычислить f(a).



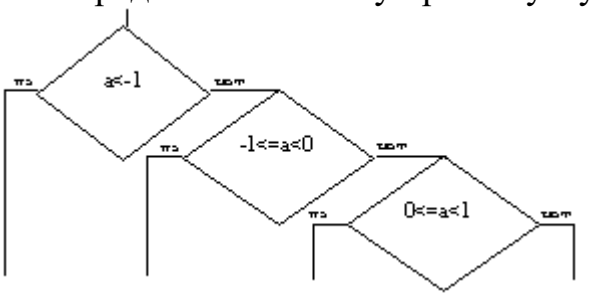
Этапы решения задачи:

1. Математическая модель: функция вычисляется по следующей формуле

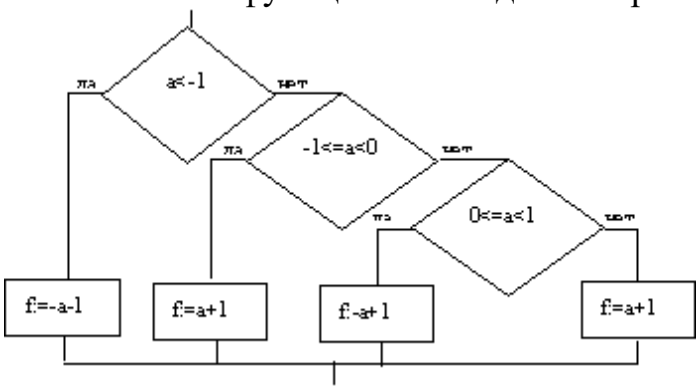
$$f(x)=\begin{cases} -x-1, & x<-1 \\ x-1, & -1\le x<0 \\ -x+1, & 0\le x<1 \\ x+1, & x\ge 1 \end{cases}$$

2. Составим схему алгоритма

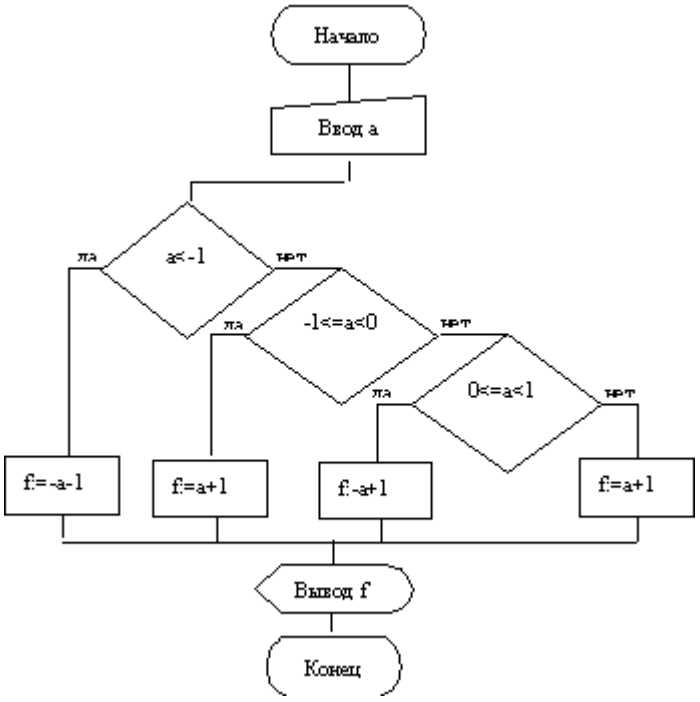
Детализируем блок"Определяем к какому промежутку относится x."



Добавим блоки вычисления функции на каждом из промежутков:



Таким образом, окончательный алгоритм выглядит следующим образом:



Переводим блок-схему на язык Паскаль.

```
Program example1;
  var a, f:real;
  begin
     write('Введите a> '); readln(a);
       if a<-1 then f:= -a-1 else
         if (a>=-1) and (a<0) then f:= a+1 else
         if (a>=0) and (a<1) then f:= -a+1 else f:= a+1;
       writeln('F= ',f);
  end.
```

Условные операторы в Си/С++

Условный оператор ***if*** используется для разветвления процесса выполнения кода программы на два направления.

В языке Си имеется две разновидности условного оператора: простой и полный. Синтаксис *простого* оператора:

***if* (*выражение*) *оператор*;**

Синтаксис *полного* оператора условного выполнения:

***if* (*выражение*) *оператор* 1 ;**
***else оператор* 2 ;**

Если ***выражение*** не равно нулю (истина), то выполняется ***оператор* 1**, иначе – ***оператор* 2**. Операторы 1 и 2 могут быть простыми или составными (блоками). Наличие символа **«;»** перед словом ***else*** в языке Си обязательно.

Оператор выбора альтернатив (переключатель)

Оператор *switch* (переключатель) предназначен для разветвления процесса вычислений на несколько направлений.

Общий вид оператора:

switch (*выражение*) {
*case константа*1: *список операторов* 1
*case константа*2: *список операторов* 2
...
case константаN: *список операторов N*
default: *список операторов N*+1 – необязательная ветвь;
}

Выполнение оператора начинается с вычисления ***выражения***, значение которого должно быть целого или символьного типа. Это значение сравнивается со значениями ***констант*** и используется для выбора ветви, которую нужно выполнить.

В данной конструкции ***константы*** фактически выполняют роль меток. Если значение выражения совпало с одной из перечисленных констант, то управление передается в соответствующую ветвь. После этого, если выход из переключателя в данной ветви явно не указан, последовательно выполняются все остальные ветви. Все константы должны иметь разные значения, но быть одного и того же типа. Несколько меток могут следовать подряд, и тогда переход в указанную ветвь будет происходить при совпадении хотя бы одной из них. Порядок следования ветвей не регламентируется.

В случае несовпадения значения выражения ни с одной из констант выбора происходит переход на метку ***default*** либо, при ее отсутствии, к оператору, следующему за оператором *switch*.

Управляющий оператор ***break*** (разрыв) выполняет выход из оператора *switch*. Если по совпадению с каждой константой должна быть выполнена одна и только одна ветвь, схема оператора switch следующая:

switch (выражение) {
case константа1: операторы 1; *break*;
case константа2: операторы 2; *break*;
...
case константа*N*: операторы *N*; *break*;
default: операторы (*N*+1); *break*;
}

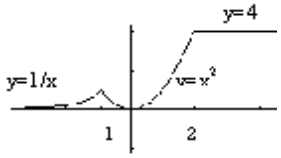
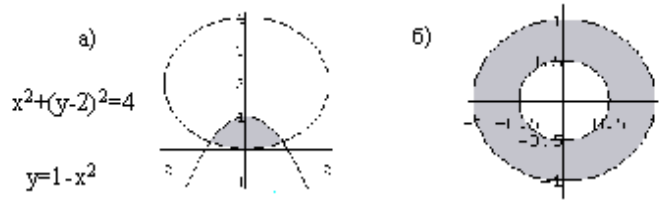
Переводим блок-схему на язык С++

```
#include <iostream>
#include <string>
#include <math.h>
using namespace std;
int main(int argc, char* argv[])
  {
   double a;
   cout<<"  a=";
   cin>>a;
    if(a<-1) cout<<"\n  f(a)="<<-a-1;
    else if((a>=-1)&&(a<0)) cout<<"\n  f(a)="<<a+1;
    else if((a>=0)&&(a<1)) cout<<"\n  f(a)="<<-a+1;
         else cout<<"\n  f(a)="<<a+1;
   }
```

Контрольные вопросы

1. Какие команды текстового редактора вы знаете?
2. Что такое блок текста и как его выделить?
3. Какие операторы используются для программирования разветвлений в Паскале?
4. Как выполняются операторы перехода?
5. Какую из функций: Sin(x), Abs(x), Trunc(x) можно заменить условным оператором if x<0 then x := -x?
6. Какие операторы используются для программирования разветвлений в Си/С++?
7. Для чего предназначен оператор *switch* (переключатель) ?

Варианты заданий на составление программ по теме «Разветвляющиеся алгоритмы»

Вариант 1	Даны действительные положительные числа x, y, z. Выяснить, существует ли треугольник с длинами сторон x, y, z.
Вариант 2	Дано действительное a. Для функции f(a), график которой представлен на рисунке, вычислить f(a). 
Вариант 3	Пусть D - заштрихованная часть плоскости и пусть u определяется по x и y следующим образом (запись (x, y)Є D означает, что точка с координатами x, y принадлежит D):  $u = \begin{cases} x - y, & \text{если } (x, y) \in D \\ xy + 7 & \text{в противном случае} \end{cases}$
Вариант 4	Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу (1, 3).
Вариант 5	Даны действительные числа x, y. Если x, y отрицательны, то каждое значение заменить его модулем; если отрицательное только одно из них, то оба значения увеличить на 0.5; если оба значения не отрицательны и ни одно из них не принадлежит отрезку [0.5, 2.0], то оба значения уменьшить в 10 раз; в остальных случаях x, y оставить без изменения.
Вариант 6	Определить и вывести на печать номер квадранта, в котором расположена точка M(x,y), x и y заданные вещественные числа.
Вариант 7	Из величин, определяемых выражениями a=sinx, b=cosx, c=ln|x| при заданном x, определить и вывести на экран дисплея минимальное значение.
Вариант 8	Определить, какая из двух точек - M1(x1,y1) или M2(x2,y2) - расположена ближе к началу координат. Вывести на экран дисплея координаты этой точки.
Вариант 9	Определить, какая из двух фигур (круг или квадрат) имеет большую площадь. Известно, что сторона квадрата равна a, радиус круга r. Вывести на экран название и значение площади большей фигуры.
Вариант 10	Составить программу нахождения значения выражения m =max(x,y,z)/min(x,y)+5 с исходными данными x,y,z.

Задачи повышенной трудности

1. Две точки заданы на плоскости своими координатами, которые могут быть как декартовыми, так и полярными. Требуется вычислить расстояние между этими двумя точками.

2. Даны действительные числа a, b, c, x, y. Выяснить, пройдет ли кирпич с ребрами a,b,c в прямоугольное отверстие со сторонами x и y. Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.

3. Сможет ли шар радиуса R пройти в ромбообразное отверстие со стороной P и острым углом Q?

4. Написать программу, которая печатает True или False в зависимости от того, выполняются или нет заданные условия:

a) квадрат заданного трехзначного числа равен кубу суммы цифр этого числа;
b) сумма двух первых цифр заданного четырехзначного числа равна сумме двух его последних цифр;
c) среди цифр заданного трехзначного числа есть одинаковые;
d) среди первых трех цифр из дробной части заданного положительного вещественного числа есть цифра 0.

5.Проверить, можно ли из четырех данных отрезков составить параллелограмм. Написать программу, определяющую попадает ли точка с координатами (x, y) в заштрихованную область

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
3. Немнюгин С. А. Turbo Pascal;практикум./МОРФ.-СПб:Питер,2003.-256с.:ил.
4. Немнюгин С.А. Turbo PASCAL:Практикум:Учебное пособие/ МОРФ.-2 изд.-М.:СПб.,2005.-267 с.
5. Культин Н. Самоучитель:Программирование в Turbo Pascal7.0 и Delphi.-2 изд.- Спб.:БХВ-Петербург,2003.-404 с
6. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
7. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
8. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.

Лабораторная работа №3
Программирование циклических алгоритмов

Цель работы: закрепить практические навыки работы с ЯП Pascal, научиться правильно использовать различные операторы циклов в C++(цикл с предусловием, цикл с постусловием и цикл с параметром); научиться составлять программы решения задач с использование циклических структур.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Большинство практических задач требует многократного повторения одних и тех же действий, т. е. повторного использования одного или нескольких операторов.

Пусть требуется ввести и обработать последовательность чисел. Если чисел всего пять, можно составить линейный алгоритм. Если их тысяча, записать линейный алгоритм можно, но очень утомительно и нерационально. Если количество чисел к моменту разработки алгоритма неизвестно, то линейный алгоритм принципиально невозможен.

Другой пример. Чтобы найти фамилию человека в списке, надо проверить первую фамилию списка, затем вторую, третью и т.д. до тех пор, пока не будет найдена нужная или не будет достигнут конец списка. Преодолеть подобные трудности можно с помощью циклов.

Циклом называется многократно исполняемый участок алгоритма (программы). Соответственно циклический алгоритм — это алгоритм, содержащий циклы.

Алгоритм называется циклическим, если он содержит многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений этих операторов может быть задано в явной (цикл с известным заранее числом повторений) или неявной (цикл с неизвестным заранее числом повторений) форме.

Различают два типа циклов: с известным числом повторений и с неизвестным числом повторений. При этом в обоих случаях имеется в виду число повторений на стадии разработки алгоритма.

Существует 3 типа циклических структур:

- Цикл с предусловием;
- Цикл с послеусловием;
- Цикл с параметром;

Иначе данные структуры называют циклами типа «Пока», «До», «Для».

Перед выполнением работы необходимо изучить различные схемы организации циклов и операторы FOR, WHILE,DO-WHILE, REPEAT.

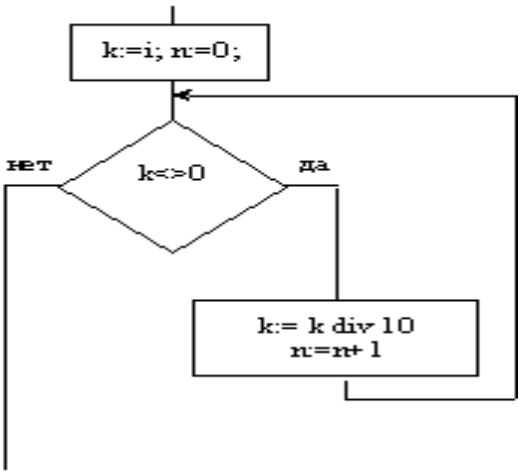
Пример: На промежутке от 1 до M найти все числа Армстронга. Натуральное число из n цифр называется числом Армстронга, если сумма его цифр, возведенных в n-ю степень, равна самому числу.

Этапы решения задачи:

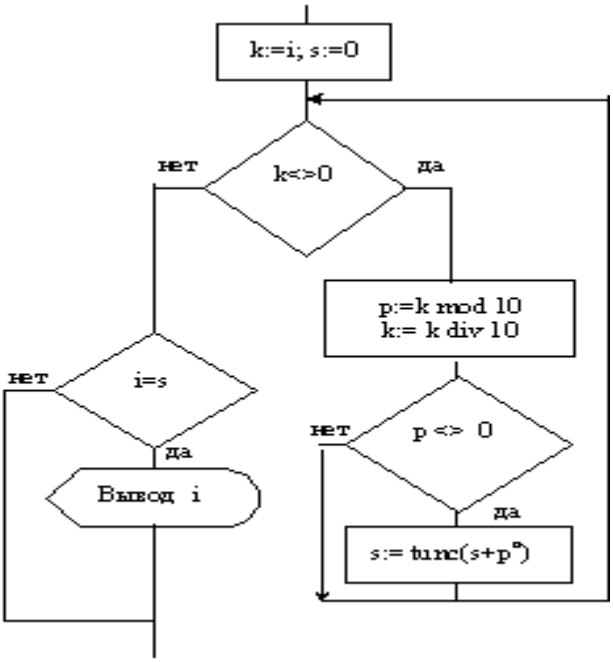
1. Составим блок схему программы:
2.

Распишем составные части блока "Находим все числа Армстронга на заданном промежутке и печатаем их"

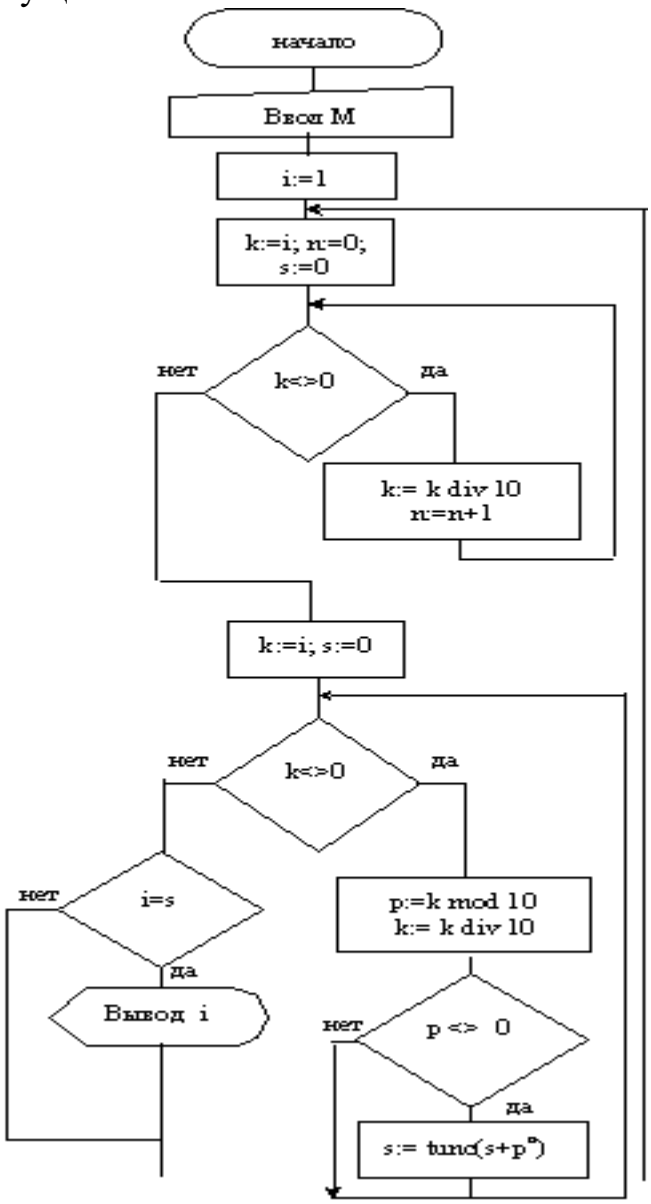
Опишем блок "Подсчитываем сколько цифр в числе i"



Опишем блок "Проверяем, является ли i числом Армстронга"



Запишем блок-схему целиком:



Переведем программу на язык Паскаль.

```
PROGRAM Primer_1;
var i,k,s,p,n: Integer;
  BEGIN
    Write('Введите M '); Readln(m);
    For i:=1 to M do
      begin
        s:=0; k:=i; n:=0;
        While k<>0 do
          begin k:=k DIV 10; n:=n+1 end;
            k:=i;
            While k<>0 do
            begin p:=k MOD 10; k:=k DIV 10;
              If p<>0 then s:=Trunc (s+Exp(n*Ln(p)))
            end;
```

```
            If s=f then WriteLn (f)
        end;
    END.
```

Оператор с предусловием while на ЯП С++

Цикл с предусловием имеет вид:

while **(***выражение***)**

код цикла;

Выражение определяет условие повторения кода цикла, представленного простым или составным оператором.

Если выражение в скобках – истина (не равно 0), то выполняется *код цикла*. Это повторяется до тех пор, пока выражение не примет значение 0 (ложь). В этом случае происходит выход из цикла и выполняется оператор, следующий за конструкцией *while*. Если выражение в скобках изначально ложно (т.е. равно 0), то цикл не выполнится ни разу.

Код цикла может включать любое количество операторов, связанных с конструкцией ***while***, которые нужно заключить в фигурные скобки (организовать блок), если их более одного.

Переменные, изменяющиеся в коде цикла и используемые при проверке условия продолжения, называются ***параметрами*** цикла. Целочисленные параметры цикла, изменяющиеся с постоянным шагом на каждой итерации, называются **счетчиками** цикла.

Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации, так и цикла в целом.Для этого используют оператор ***continue*** – переход к следующей итерации цикла и ***break*** – выход из цикла.

Передавать управление извне внутрь цикла не рекомендуется, так как получите непредсказуемый результат.

Оператор цикла с постусловием do – while на ЯП С++

Общий вид записи такой конструкции

do

код цикла;

while (*выражение*);

Код цикла будет выполняться до тех пор, пока *выражение* истинно. Все, что говорилось выше, справедливо и здесь, за исключением того, что данный цикл всегда выполняется хотя бы один раз, даже если изначально выражение ложно. Здесь сначала выполняется код цикла, после чего проверяется, надо ли его выполнять еще раз.

Оператор цикла с предусловием и коррекцией for на ЯП С++

Общий вид оператора:

for (*выражение* 1; *выражение* 2; *выражение* 3)

код цикла;

где *выражение* 1 – инициализация счетчика (параметр цикла);

выражение 2 – условие продолжения счета;

выражение 3 – коррекция счетчика.

Инициализация используется для присвоения счетчику (параметру цикла) начального значения.

Выражение 2 определяет условие выполнения цикла. Как и в предыдущих случаях, если его результат не нулевой («истина»), – то цикл выполняется, иначе – происходит выход из цикла.

Коррекция выполняется после каждой итерации цикла и служит для изменения параметра цикла.

Выражения 1, 2 и 3 могут отсутствовать (пустые выражения), но символы «;» опускать нельзя.

Операторы continue, break и return на ЯП С++

Оператор ***continue*** может использоваться во всех типах циклов (но не в операторе-переключателе *switch*). Наличие оператора *continue* вызывает пропуск «оставшейся» части итерации и переход к началу следующей, т.е. досрочное завершение текущего шага и переход к следующему шагу.

В циклах *while* и *do-while* это означает непосредственный переход к проверочной части. В цикле *for* управление передается на шаг коррекции, т.е. модификации *выражения* 3.

Оператор *continue* часто используется, когда последующая часть цикла оказывается слишком сложной, так что рассмотрение условия, обратного проверяемому, приводит к слишком высокому уровню вложенности программы.

Оператор ***break*** производит досрочный выход из цикла или оператора-переключателя *switch*, к которому он принадлежит, и передает управление первому оператору, следующему за текущим оператором. То есть *break* обеспечивает переход в точку кода программы, находящуюся за оператором, внутри которого он (*break*) находится.

Оператор ***return*** производит досрочный выход из текущей функции. Он также возвращает значение результата функции:

return *выражение*;

Выражение должно иметь скалярный тип.

Переводим блок-схему на язык С++.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  int figure = 0, summa = 0;
  int  i, a, b, st, n;
```

```
cout<<"Programma vivodit vse chisla Armstronga ot 1 do n"<<endl;
cout<<"Vvedite n= "; cin>>n;
for ( i = 1; i < n; i++)
{
   a = b = i;
   while(a)
   {
      a /= 10;
      figure++;
   }
   st = pow(n, figure-1);
   while(b)
   {
      summa += pow((double)(b / st), figure);
      b %= st;
      st /= 10;
   }
   if(summa == i)
   cout << i << "  Armstrong number " << endl;
   figure = 0;
   summa = 0;
  }
 }
```

Контрольные вопросы

1. Как записывается и как работает оператор FOR?
2. Для организации каких циклов применим оператор FOR?
3. В чем отличие оператора WHILE от оператора REPEAT?
4. Как программируются циклические алгоритмы с явно заданным числом повторений цикла?
5. Как программируются циклические алгоритмы с заранее неизвестным числом повторений цикла?
6. Напишите оператор цикла, который не выполняется ни разу.
7. Напишите оператор цикла, который выполняется неограниченное число раз.

Варианты заданий для составления программ по теме «Циклические алгоритмы»

Вариант 1	Дано натуральное число n. Вычислить $(1+\frac{1}{1^2})+(1+\frac{1}{2^2})+(1+\frac{1}{3^2})+\ldots+(1+\frac{1}{n^2})$;
Вариант 2	Дано действительное число x, натуральное число n. Вычислить:

	$\frac{1}{x}+\frac{1}{x(x+1)}+\ldots+\frac{1}{x(x+1)\ldots(x+n)}$;
Вариант 3	Дано действительное число x, натуральное число n. Вычислить: $\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}$;
Вариант 4	Дано натуральное число n. Вычислить : $\prod_{i=1}^{n}(2+1/i!)$
Вариант 5	Дано натуральное число n. Вычислить : $\sum_{i=1}^{n}(1+i)/i!$;
Вариант 6	Вычислить приближенно значение бесконечной суммы (справа от каждой суммы дается ее точное значение, с которым можно сравнить полученный ответ): $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots=\frac{\pi^2}{6}$.
Вариант 7	Вычислить приближенно значение бесконечной суммы (справа от каждой суммы дается ее точное значение, с которым можно сравнить полученный ответ): $\frac{1}{1\cdot3}+\frac{1}{2\cdot4}+\frac{1}{3\cdot5}+\ldots=\frac{3}{4}$;
Вариант 8	Вычислить приближенно значение бесконечной суммы (справа от каждой суммы дается ее точное значение, с которым можно сравнить полученный ответ): $1+\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots=e^x$;
Вариант 9	Можно ли заданное натуральное число M представить в виде суммы квадратов двух натуральных чисел?
Вариант 10	Дано натуральное число n. Вычислить $\frac{1}{\sin1}+\frac{1}{\sin1+\sin2}+\ldots+\frac{1}{\sin1+\ldots\sin n}$

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
3. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.

Лабораторная работа 4
Программирование с использованием массивов

Цель работы: научиться правильно описывать различные массивы, уметь инициализировать массивы, распечатывать содержимое массива; научится решать задачи на использование массивов.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal, Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения:

Массив — упорядоченный набор данных, для хранения данных одного типа, идентифицируемых с помощью одного или нескольких индексов. В простейшем случае массив имеет постоянную длину и хранит единицы данных одного и того же типа. . Для обозначения элементов массива используются имя переменной-массива - индекс

Количество используемых индексов массива может быть различным. Массивы с одним индексом называют одномерными, с двумя — двумерными и т. д. Одномерный массив (колонка, столбец) нестрого соответствует вектору в математике, двумерный — матрице. Чаще всего применяются массивы с одним или двумя индексами, реже — с тремя, ещё большее количество индексов встречается крайне редко.

В языках программирования, допускающих объявления программистом собственных типов, как правило, существует возможность создания типа «массив». В определении такого типа может указываться размер, тип элемента, диапазон значений и типы индексов. В дальнейшем возможно определение переменных созданного типа. Все такие переменные-массивы имеют одну структуру. Некоторые языки поддерживают для переменных-массивов операции присваивания (когда одной операцией всем элементам массива присваиваются значения соответствующих элементов другого массива).

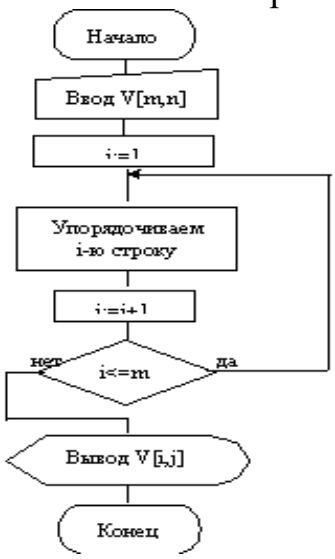
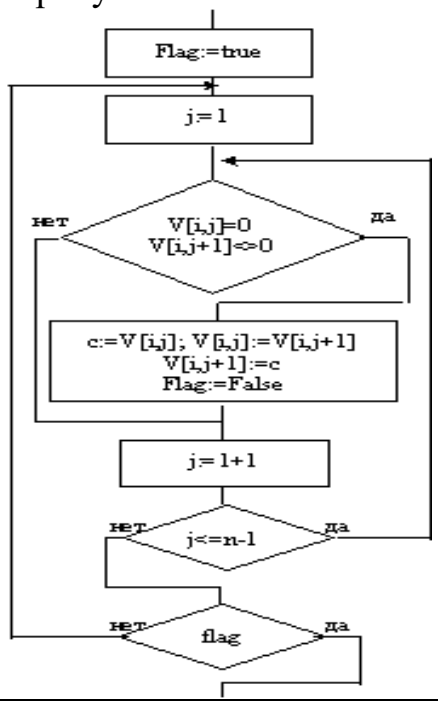
Перед выполнением работы необходимо изучить правила описания и использования переменных типа массив, типизированных констант типа массив.

Пример: Дан двумерный массив. В каждой строке все его элементы, не равные нулю, переписать (сохраняя порядок) в начало строки, а нулевые элементы - в конец массива. Новый массив не заводить.

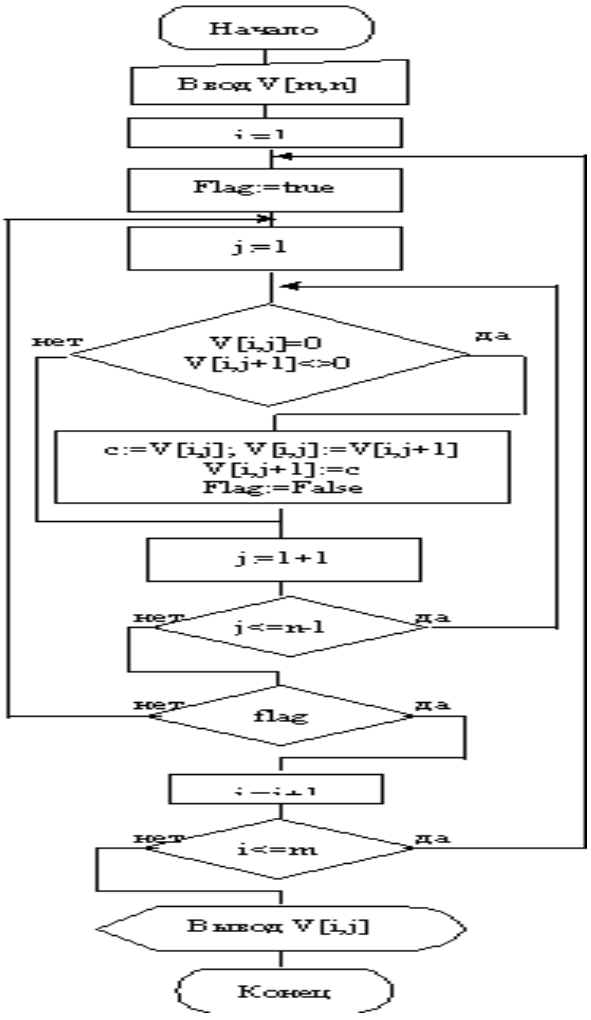
Этапы решения задачи:

1. Суть одного из алгоритмов решения данной задачи состоит в том чтобы "просматривать" массив построчно и находить в каждой строке пару (0:число), а затем менять их местами между собой и так до тех пор пока в строке таких пар не окажется.

2.Составим блок схему

блок –схема алгоритма	Детализируем блок "Упорядочиваем 1-ю строку"
	

Блок схема алгоритма целиком:



3.Приведем программу на языке Паскаль:

```
program example1;

  var
    V:array[1..100,1..100] of integer;
     m,n, i,j, c: integer;
       flag: boolean;
begin
      write('Введите размерность массива m-n> '); readln(m,n);
        for i:= 1 to m do
          for j:= 1 to n do begin
            write('V[',i,',',j,']= '); readln(V[i,j]);
          end;
    for i:=1 to m do
      repeat
         flag:= true;
           for j:=1 to n-1 do
        if (v[i,j]=0) and (v[i,j+1]<>0) then begin
        c:=v[i,j]; v[i,j]:=v[i,j+1]; v[i,j+1]:=c; flag:= false;
        end;
      until flag;
      for i:= 1 to m do begin
        for j:= 1 to n do write(V[i,j]:2);
        writeln
     end;
  readln;
end.
```

Приведем программу на языке C++

```
#include    <iostream>
#include    <vector>
int main(){
   std::vector<int> mas;
   int n, temp;
   std::cout << "Razmernost      massiva
n:\t";
   std::cin >> n;
   std::cout << "\n\n";
   for(int k=0; k<n; k++){
       std::cout << "Wvedite element:\t";
       std::cin >> temp;
       mas.push_back(temp);
    }
temp = n;
    for(int i=0; i<n; i++){
        if(mas[i] == 0){
            for(int j=i; j<n-1; j++){
                mas[j] = mas[j+1];
            }
            n--;
        }
    }
    for(int k=0; k<temp; k++){
        std::cout << mas[k] << " ";
    }
        return 0;
}
```

Контрольные вопросы

1. Каким образом объявляется массив (одномерный и двумерный)на ЯП СИ/С++ ?
2. Как осуществляется доступ к отдельному элементу одномерного и двумерного массива?
3. Каким образом выводятся элементы массива на экран?
4. В чем отличие при задании массивов на Си от Паскаля?
5. С какой цифры начинаются индексы массивов в языке Си?

Задания для составления программ по теме «Массивы»

Вариант 1	Даны целые числа a1, a2, a3. Получить целочисленную матрицу [bij]i,j=1,2,3, для которой bij=ai-3aj.
Вариант 2	Дано натуральное число n. Получить действительную матрицу [aij]i, j=1,…n, для которой aij=i+2j.
Вариант 3	Дана квадратная вещественная матрица размерности n. Найти количество нулевых элементов, стоящих: выше главной диагонали; ниже главной диагонали; выше и ниже побочной. а) б) в) г) 
Вариант 4	Дана вещественная матрица размерности n * m. По матрице получить логический вектор, присвоив его k-ому элементу значение True , если выполнено указанное условие и значение False иначе: - все элементы k столбца нулевые; - элементы k строки матрицы упорядочены по убыванию; - k строка массива симметрична.
Вариант 5	Дана вещественная матрица размерности n * m. Сформировать вектор b, в котором элементы вычисляются как: - произведение элементов соответствующих строк; - среднее арифметическое соответствующих столбцов; - разность наибольших и наименьших элементов соответствующих строк;
Вариант 6	Дан двухмерный массив A[1..m,1..n]. Написать программу построения одномерного массива B[1..m], элементы которого соответственно равны а) суммам элементов строк, б) произведениям элементов строк,.
Вариант 7	Расположить элементы данного массива в обратном порядке (первый элемент меняется с последним, второй - с предпоследним и т.д. до середины; если массив содержит нечетное количество элементов, то средний остается без изменения).
Вариант 8	В данном массиве поменять местами элементы, стоящие на нечетных местах, с элементами, стоящими на четных местах.
Вариант 9	Даны целые числа a1, a2, a3. Получить [aij], i=1,…10; j=1,…12 - целочисленную матрицу, для которой aij=i+2j.

Вариант 10	Дан двухмерный массив B[1..m,1..n]. Написать программу построения одномерного массива A[1..m], элементы которого соответственно равны наименьшим средних арифметических элементов строк.

Задачи повышенной сложности

1. В массиве A[1..N,1..N] определить номера строки и столбца какой-нибудь седловой точки. Некоторый элемент массива называется седловой точкой, если он является одновременно наименьшим в своей строке и наибольшим в своем столбце.
2. Массив A[1..5,1..7] содержит вещественные числа. Требуется ввести целое число K и вычислить сумму элементов A[I,J], для которых I+J=K. Прежде, однако следует убедиться, что значение K позволяет найти решение, в противном случае нужно напечатать сообщение об ошибке.
3. Дан массив A[1..N,1..N]. Составить программу, которая прибавила бы каждому элементу данной строки элемент, принадлежащий этой строке и главной диагонали.
4. Дана матрица NxM. Переставляя ее строки и столбцы, переместить наибольший элемент в верхний левый угол. Определить можно ли таким же образом поместить минимальный элемент в нижний правый угол.
5. Заполнить двухмерный массив T[1..n,1..n] последовательными целыми числами от 1 до n 52 0, расположенными по спирали, начиная с левого верхнего угла и продвигаясь по часовой стрелке:

Литература основная

1. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
2. Немнюгин С.А. Turbo PASCAL:Практикум:Учебное пособие/ МОРФ.-2 изд.-М.:СПб.,2005.-267 с.
3. Культин Н. Самоучитель:Программирование в Turbo Pascal7.0 и Delphi.-2 изд.- Спб.:БХВ-Петербург,2003.-404 с
4. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
5. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
6. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.
7. Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.

Лабораторная работа 5
Строковый тип данных.
Обработка символьных строк в С++.

Цель работы: использование строкового типа данных "string" в Turbo Pascal, выработка навыков работы с символьной информацией в языке программирования Си/С++; научиться использовать строки символов и множества при решении задач.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

В программировании, строковый тип — тип данных, значениями которого является произвольная последовательность (строка) символов алфавита. Каждая переменная такого типа (строковая переменная) может быть представлена фиксированным количеством байтов либо иметь произвольную длину.

Переменные типа String аналогичны массивам типа Char. Их отличием является то, что число символов (длина строки) может динамически меняться в интервале от единицы до заданного верхнего значения.

В языках программирования Си/С++ нет определенного строкового типа данных.Символьные строки организуются как массивы символов, на длину символьного массива в Си нет ограничения.

В этом подходе строки представляются массивом символов; при этом размер массива хранится в отдельной (служебной) области.

Перед выполнением работы необходимо ознакомиться с правилами описания и использования строк, допустимых операций над ними, соответствующими стандартными процедурами и функциями.

Пример. Дан текст, слова в котором, могут разделяться пробелами, запятыми, точками и т.д. Требуется напечатать все слова с удвоенной буквой "н".

Этапы решения задачи:

1. Разобьем задачу на несколько блоков а) Формирование тела программы, объявление переменных; б) Ввод текста; в) Очистка текста от "ненужных" символов до первого слова; г) Вычисление длины первого слова; д) Поиск в слове буквы "н"; е) Подсчет стоящих рядом букв "н"; ж) Печать найденного слова; з) Удаление первого слова; и) Если текст не закончился возвращение к пункту (в).
2. Реализуем эти блоки на Паскале

а)

```
program example1;
 var st, st1:string;
        i,j,k,n:integer;
```

```
     flag:boolean;
    const
  znak=[' ',',','.',':',';','!','?'];
 begin
end.
```

Назначение переменных:t- содержит введенный текст
st1 - хранит первое слово текста
i,j,k,n - вспомогательные переменные
flag - указывает, что данное слово искомое

```
б)        writeln('Введите текст'); readln(st);
в)        repeat
            while st[1] in znak do delete(st,1,1);
г) i:=1  while (not (st[i] in znak)) and (i<=length(st)) do inc(i);
          st1:=copy(st,1,i-1);
          flag:= false;
д)   while (pos('н',st1)>0) and (not flag) do begin
е)   j:=pos('н',st1); n:=j; k:=0;
     while st1[n]='н' do begin inc(n); inc(k); end;
     if k=2 then flag:= true;
     delete(st1,j,k)
     end;
ж)   if flag then writeln(copy(st,1,i-1));
з)   delete(st,1,i);
и)   until st='';
```

Приведем программу целиком:

```
program example1;
var st, st1:string;
      i,j,k,n:integer;
      flag:boolean;
const
    znak=[' ',',','.',':',';','!','?'];
  begin
    writeln('Введите текст');
 readln(st);
    repeat
        while st[1] in znak do delete(st,1,1);
      i:=1;
    while (not (st[i] in znak)) and (i<=length(st)) do inc(i);
    st1:=copy(st,1,i-1);
    flag:= false;
    while (pos('н',st1)>0) and (not flag) do begin
     j:=pos('н',st1); n:=j; k:=0;
        while st1[n]='н' do begin inc(n); inc(k); end;
```

```
      if k=2 then flag:= true;
      delete(st1,j,k)
  end;
  if flag then writeln(copy(st,1,i-1));
    delete(st,1,i);
   until st='';
    readln;
end.
```

Код программы на C++:

```
#include <iostream>
#include <cstring>
void insert(char* buffer, unsigned pos, char c)
{
    for(unsigned i = strlen(buffer) + 1; i > pos; --i)
        buffer[i] = buffer[i - 1];
    buffer[pos] = c;
}
 int main()
{
    std::cout << "vvedite stroky: ";
    const int BUF_SIZE = 256;
    char* buffer = new char[BUF_SIZE];
    std::cin.getline(buffer, BUF_SIZE);
     std::cout << "wvedite bykvy: ";
    char c;
    std::cin >> c;
    for(unsigned i = 0; i < strlen(buffer); ++i)
        if(c == buffer[i])
        {
            insert(buffer, i, c);
            ++i;
        }
     std::cout << buffer << std::endl;
    delete[] buffer;
    return 0;
}
```

Контрольные вопросы

1. Как описываются строковые переменные?
2. Какая максимальная длина строки допустима в Pascal, C++?
3. Какие операции допустимы над строковыми данными?
4. В чем отличие строковой переменной от массива символов?

5. Какие стандартные процедуры и функции для работы со строками вы знаете?
6. Что выведет функция Copy(x,Pos(' ',x)+1,18), если x='Сила есть - ума не надо'?
7. Как реализуется работа со строками в языке Си?

Задания для составления программ по теме « Обработка символьных строк»

Обработка текста: В следующих заданиях под словом "текст" понимается строка символов, слова в которой, разделены пробелами, ",",".","!","?",";",":" (одним или несколькими).

Вариант 1	Дан текст. а) Подсчитать количество слов в данной строке. б) Подсчитать количество букв а в последнем слове данной строки. в) Найти количество слов, начинающихся с буквы б. г) Найти количество слов, у которых первый и последний символы совпадают между собой. д) Найти длину самого короткого слова.
Вариант 2	Составить программу циклической перестановки букв в словах текста так, что i-я буква слова становится i+1-ой, а последняя - первой.
Вариант 3	В каждом слове текста замените "а" на букву "е", если "а" стоит на четном месте, и заменить букву "б" на сочетание "ак", если "б" стоит на нечетном месте.
Вариант 4	Гжатск получил новое название - город Гагарин. А в рязанской областной типографии еще не просохли гранки небольшой книги о родине первого космонавта. Конечно, книгу нужно было переделать... Написать программу, осуществляющую в некотором тексте замену слова "Гжатск" словом "Гагарин" (учесть, что слова имеют разную длину!)
Вариант 5	Дан текст, содержащий от 2 до 30 слов, в каждом из которых от 2 до 10 латинских букв; между соседними словами - не менее одного пробела. Напечатать все слова, отличные от последнего слова, предварительно преобразовав каждое из них по следующему правилу: 1) перенести первую букву в конец слова; 2) перенести последнюю букву в начало слова.
Вариант 6	Отредактировать заданное предложения текста, удаляя из него все слова с нечетными номерами и переворачивая слова с четными номерами. Например, HOW DO YOU DO -> OD OD
Вариант 7	Дан текст. Напечатать все слова, отличные от последнего слова, предварительно преобразовав каждое из них по следующему правилу: 1) оставить в слове только первые вхождения каждой буквы; 2) если слово нечетной длины, то удалить его среднюю букву
Вариант 8	Написать программу для подсчета суммы мест, на которых в словах текста стоит заданная буква.

Вариант 9	Составить таблицу слов данного текста, начинающихся с буквы "А", с указанием числа повторений каждого слова.
Вариант 10	Составить программу для вычеркивания из слов текста всех букв, стоящих на нечетных местах после буквы "а". Задачи на смекалку
Вариант 11	Составить программы для перевода арабских чисел в римские и для обратной операции. Например, 255 = CCLV = сто + сто + пятьдесят + пять. Замечание. Подобными алгоритмами перевода чисел из одной системы в другую мы пользуемся по нескольку раз на дню, когда ведем денежные расчеты. Сумма денег - это арабское число, которому соответствует определенный набор банкнот и монет (аналоги римских цифр).
Вариант 12	Автоморфными называются числа, которые содержатся в последних разрядах их квадрата. Например:, 52=25, 252=625. Составить программу для нахождения нескольких автоморфных чисел.
Вариант 13	Подсчитать, сколько букв надо исправить в слове X, чтобы получилось слово Y (X,Y - слова одинаковой длины).
Вариант 14	Какое минимальное число букв необходимо заменить в слове X с тем, чтобы оно стало перевертышем?
Вариант 15	Составить программу для подсчета числа одинаковых букв в словах X и Y равной длины, стоящих на одних и тех же местах.
Вариант 16	Задано определенное количество конкретных сочетаний букв (например, УЩ, ЮЩ и др.). Определить, сколько таких групп символов содержится в тексте, вводимом с клавиатуры.
Вариант 17	С клавиатуры вводится текст. Подсчитать и вывести на печать количество слов текста, начинающихся с гласной.
Вариант 18	Для запоминания числа ПИ иногда используют "магические" фразы, например: "это я знаю и помню прекрасно Пи многие знаки мне лишни напрасны"и "кто и шутя и скоро пожелаетъ Пи узнать число ужъ знаетъ". Число букв в каждом слове любой из данных фраз представляет собою некоторую цифру числа : "это"-3, "я"-1, "знаю"-4 и т.д. Составить программу, которая по указанному алгоритму будет выводить на печать число, используя любой текст.
Вариант 19	Для заданного текста определить длину содержащейся в нем максимальной серии символов, отличных от латинских букв.
Вариант 20	Записать программу, выясняющую, можно ли из букв слова X составить слово Y.

Задачи повышенной сложности

1. Зашифровать введенную с клавиатуры строку, поменяв местами первый символ со вторым, третий с четвертым и т. д. Затем провести дополнительную шифровку результата смещением кода. Провести дешифровку.

2. Составить процедуру создания текстового окна, окаймленного рамкой из псевдографических символов. В параметры процедуры ввести координаты левого верхнего угла, размеры и цвет окна, а также цвет рамки.

3. Составить программу, организующую перемещение текстового окна 8x8 по экрану. См. задачу 2. Движение начинается по нажатию клавиши и заканчивается либо по нажатию клавиши, либо при достижении окном края экрана. Варианты движения: а) из левого верхнего угла в правый нижний угол. При неточном "попадании" в нижний угол смещать окно по одной из сторон до точной остановки в углу. б)из левого нижнего угла в правый верхний. в) из центра экрана к одной из боковых сторон. При достижении края размер окна по направлению движения должен уменьшаться до минимального.

Литература основная

1. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
2. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
3. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
4. Архангельский, А. Я. Программирование в C++ Builder 6 / А. Я. Архангельский. – М. : ЗАО «Издательство БИНОМ», 2002.
5. Березин, Б. И. Начальный курс C и C++ / Б. И. Березин, С. Б. Березин. – М. : Диалог– МРТИ, 1999.
6. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. – СПб. : Невский диалект, 2001.
7. Вирт, Н. Алгоритмы + структуры данных = программы / Н. Вирт. – М. : Мир, 1985.
8. Касаткин, А. И. Профессиональное программирование на языке Си: От Turbo-C к Borland C++: справ.пособие / А. И. Касаткин, А. Н. Вольвачев. – Минск : Выш. шк., 1992.

Лабораторная работа 6
Программирование с использованием множеств

Цель работы: закрепление навыков программирования с множественным типом данных в Pascal , знакомство с понятием *"enum"* в языке программирования C++; выработать навыки работы со структурированным типом данных .

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Под множеством понимают ограниченный, неупорядоченный набор различных элементов одного типа. В отличие от массивов к элементам множества нет прямого доступа (по индексам этих элементов, как в массивах). Поэтому ввод-вывод множеств производится с использованием операций объединения (при вводе) и проверки принадлежности (при выводе). Под мощностью множества понимают количество элементов, содержащихся в данном множестве.

Множество — тип и структура данных в информатике, является реализацией математического объекта множество.

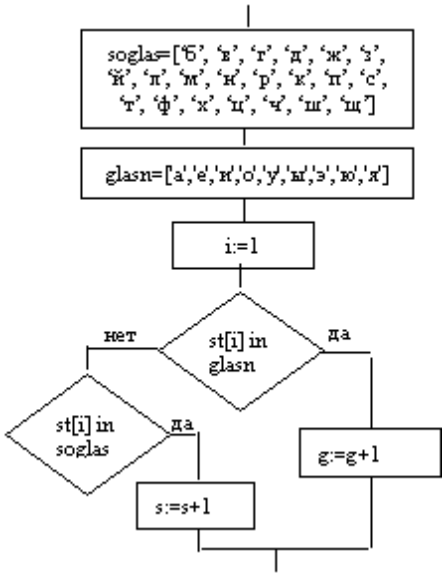
Данные типа множество позволяют хранить ограниченное число значений определённого типа без определённого порядка. Повторение значений, как правило, недопустимо. За исключением того, что множество в программировании конечно, оно в общем соответствует концепции математического множества. Для этого типа в языках программирования обычно предусмотрены стандартные операции над множествами. В зависимости от идеологии, разные языки программирования рассматривают множество как простой или сложный тип данных.

Перед выполнением работы необходимо ознакомится с правилами описания и использования переменных типа множество, типизированных констант типа множество, переменных, заданных перечислением, изучить допустимые операции над переменными этих типов.

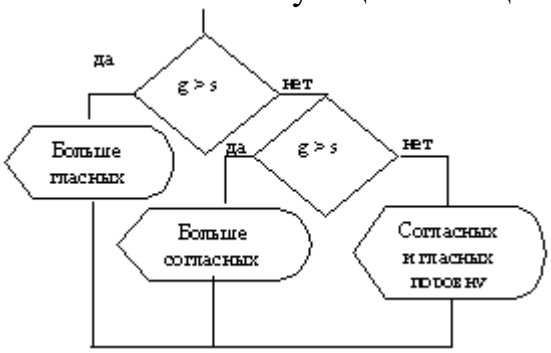
Пример: Дан текст. Определить каких букв больше - гласных или согласных.

Этапы решения задачи: 1. Составим блок схему программы

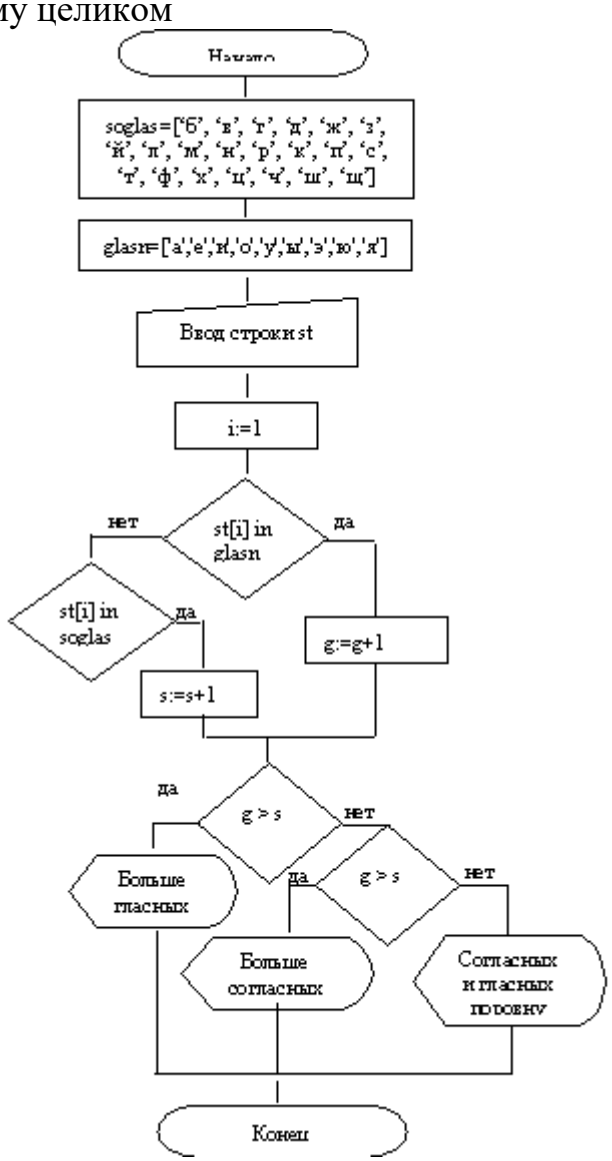
Опишем подробнее блок "Подсчитываем количество гласных и согласных букв"



Рассмотрим блок "Печатаем соответствующее сообщение"



Запишем блок-схему целиком



2. Переведем алгоритм на язык Паскаль

```
program example1;
const
  glasn=['а','е','и','о','у','ы','э','ю','я'];
   soglas=['б','в','г','д','ж','з','й','л','м', 'н','р','к','п','с','т','ф','х','ц','ч','ш','щ'];
 var
   st: string;
     g,s,i:integer;
begin
  write('Введите строку> '); readln(st);
    g:=0; s:=0;
      for i:= 1 to length(st) do
        if st[i] in glasn then inc(g) else if st[i] in soglas then inc(s);
        if g> s then writeln('Гласных больше')
    else if g< s then writeln('Согласных больше')
  else writeln('Согласных и гласных букв поровну');
 readln; end.
```

Переведем алгоритм на язык C++:

```
#include <iostream>
#include <string.h>
using namespace std ;
int main(void)
 {
int s=0,k=0,i,j;
char st[30];
const
char glasn[]="aoiyeu",
soglas[]="bpvfdtzsjgkhmnlr";
cout<<"VVedite stroky"<< endl;
cin.getline(st,30);
cout<<st;
cout<<" ";
    for(i=0;st[i]!=0;i++)
    {
    for(j=0;glasn[j]!=0;j++)
     {
      if(st[i]==glasn[j])
       k++;
     }
    for(j=0;soglas[j]!=0;j++)
     {
      if(st[i]==soglas[j])
       s++;
     }
    }
 cout<<"\n";
 if (k> s)
  cout <<" Glasn >";
 else
  if (k==s)
   cout <<"Porovny";
  else
   cout <<"Soglas>";
return 0;
}
```

Контрольные вопросы

1.Что такое множество, как оно описывается в языке Pascal?

2.Как определить новый тип данных с использованием перечисления?

3.Как описываются типизированные константы типа множество?

4.Как осуществляется ввод-вывод значений переменных типа множество?

5.Какие типы данных используются в качестве базовых при объявлении типа множество?

6.Какие операции определены над множествами?

7.Какие операции допустимы над переменными, заданными перечислением?

8.Чем похожи и чем отличаются множества и массивы?

Задания для составления программ

Примечание: Гласные буквы - а,е,и,о,у,ы,э,ю,я (ё обычно не входит в литерный тип); согласные - все остальные буквы, кроме ь, ъ; звонкие согласные - б,в,г,д,ж,з,й,л,м,н,р; глухие согласные - к,п,с,т,ф,х,ц,ч,ш,щ.

Вариант 1	Дан текст из строчных латинских букв, за которым следует точка Напечатать: - первые вхождения букв в текст, сохраняя их взаимный исходный порядок; все буквы, входящие в текст не менее двух раз; все буквы, входящие в текст по одному разу. Дана непустая последовательность слов из строчных русских букв; между соседними словами - запятая, за последним словом - точка.
Вариант 2	Напечатать в алфавитном порядке: все гласные буквы, которые входят в каждое слово; все согласные буквы, которые не входят ни в одно слово;
Вариант 3	Напечатать в алфавитном порядке все звонкие согласные буквы, которые входят хотя бы в одно слово; все глухие согласные буквы, которые не входят хотя бы в одно слово;
Вариант 4	Напечатать в алфавитном порядке все согласные буквы, которые входят только в одно слово; все глухие согласные буквы, которые не входят только в одно слово;
Вариант 5	Напечатать в алфавитном порядке все звонкие согласные буквы, которые входят более чем в одно слово; все гласные буквы, которые не входят более чем в одно слово;
Вариант 6	Напечатать все звонкие согласные буквы, которые входят в каждое нечетное слово и не входят ни в одно четное слово; все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово.
Вариант 7	Имеются три множества символьного типа, которые заданы своими конструкторами: Y1=['A','B','D','R','H'] Y2=['R','A','H','D'] Y3=['A','R'].

	Сформировать новое множество.
Вариант 8	Подсчитать общее количество цифр и знаков '+', '-', и '*', входящих в строку s.
Вариант 9	Подсчитать количество различных (значащих) цифр в десятичной записи натурального числа n и напечатать в возрастающем порядке все цифры, не входящие в десятичную запись натурального числа n.
Вариант 10	Вычислить сумму тех элементов матрицы A, номера строк и столбцов которых принадлежат соответственно непустым множествам S1 и S2 типа Nom. Cons n=10; Type Nomer= 1..n; Матрица = Array[Nomer,Nomer] of Real; Nom = Set of Номер;

Задачи повышенной сложности

1. Составить программу ,печатающую в возрастающем порядке все целые числа из диапазона 1..255,представимые в виде n^2+m^2 ,где n,m>0
2. Написать программу раздачи карт при игре в дурака, количество игроков задается с клавиатуры.
3. Для произвольного символьного множества сгенерировать все подмножества

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
3. Немнюгин С. А. Turbo Pascal;практикум./МОРФ.-СПб:Питер,2003.-256с.:ил.
4. Немнюгин С.А. Turbo PASCAL:Практикум:Учебное пособие/ МОРФ.-2 изд.-М.:СПб.,2005.-267 с.
5. Культин Н. Самоучитель:Программирование в Turbo Pascal7.0 и Delphi.-2 изд.- Спб.:БХВ-Петербург,2003.-404 с
6. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
7. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
8. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
9. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.

Лабораторная работа 7
Программирование с использованием типа запись Структуры

Цель работы: закрепить навыки программирования в Pascal с понятием "запись"-record, умения инициализировать переменные типа record, выводить на экран переменные типа record; выработать навыки работы со структурным типом данных в языке программирования C++, научиться правильно использовать тип struct .

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Под записью понимается структура данных, объединяющая под одним именем данные различных типов. Записи состоят из фиксированного числа элементов, называемых полями. Поле - это переменная определенного типа. Различные поля могут иметь (в отличие от элементов массива) различный тип. Обращение к элементу записи выполняется с помощью составного имени. Первая часть составного имени - это имя записи, вторая часть - имя поля. Между именем записи и именем поля ставится точка: ZAP.IMP, где ZAP - имя записи, IMP- имя поля.

Перед выполнением работы необходимо изучить правила описания и использования записей, типизированных констант типа запись, оператора присоединения WITH.

В реальных задачах информация, которую требуется обрабатывать, может иметь достаточно сложную структуру. Для ее адекватного представления используются типы данных, построенные на основе базовых типов данных, массивов и указателей. Языки высокого уровня позволяют программисту определять свои типы данных и правила работы с ними, т.е. типы, определяемые пользователем. В языке Си к ним относятся структуры, объединения и перечисления. Рассмотрим их более подробно.

Структуры

Структура – это составной объект языка Си, представляющий собой совокупность логически связанных данных различных типов, объединенных в группу под одним идентификатором. Данные, входящие в эту группу, называют полями.

Термин «***структура***» в языке Си соответствует двум разным по смыслу понятиям:

– структура – это обозначение участка оперативной памяти, где располагаются конкретные значения данных; в дальнейшем – это

структурная переменная, поля которой располагаются в смежных областях ОП;

– структура – это правила формирования структурной переменной, которыми руководствуется компилятор при выделении ей места в ОП и организации доступа к ее полям.

Определение объектов типа структуры производится за два шага:

– декларация структурного типа данных, не приводящая к выделению участка памяти;

– определение структурных переменных объявленного структурного типа с выделением для них памяти.

Декларация структурного типа данных

Структурный тип данных задается в виде шаблона, общий формат описания которого следующий:

```
struct ID структурного типа {
        описание полей
};
```

Атрибут «*ID структурного типа*» является необязательным и может отсутствовать. Описание полей производится обычным способом: указываются типы переменных и их идентификаторы.

Пример определения структурного типа

Необходимо создать шаблон, описывающий информацию о студенте: номер группы, Ф.И.О. и средний балл. Один из возможных вариантов:

```
struct Stud_type  {
        char Number[10];
        char Fio[40];
        double S_b;
};
```

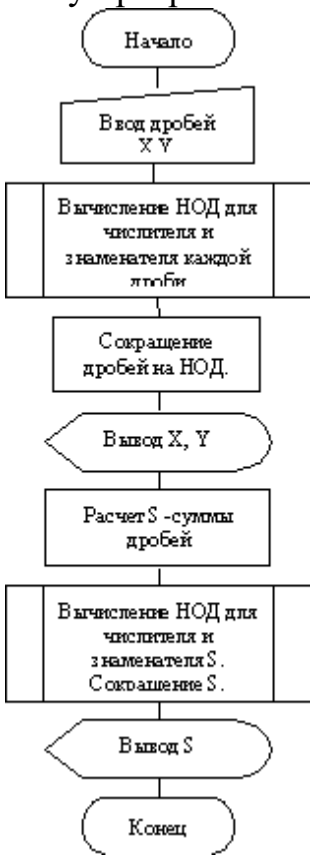
Поля одного типа при описании можно объединять в одну группу:

```
struct Stud_type  {
        char Number[10], Fio[40];
        double S_b;
};
```

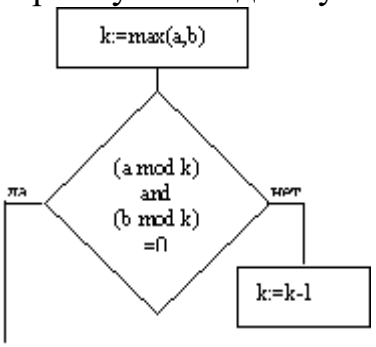
Пример. Даны два рациональных числа, опишите их, используя структуру данных запись (числитель, знаменатель). Привести их к несократимому виду, найди их сумму.

Этапы решения программы:

1. Составим блок-схему программы



2. Уточним содержимое блока "Вычисление НОД для числителя и знаменателя каждой дроби".
Во-первых, НОД это наибольший общий делитель, число крайне необходимое чтобы сократить дробь.
Вычислять НОД будем по алгоритму Евклида. Пусть даны два числа a и b:



В качестве чисел a и b будут участвовать числитель и знаменатель дробей.
Как сокращать и складывать дроби, надеюсь, вы еще не забыли, поэтому детализировать этот блок не будем. Желательно блок "Поиск НОД" оформить в виде функции, что будет предварять тему следующей лабораторной работы.

3. Переведем программу на язык Паскаль

```
PROGRAM Example1;
  Type Tfraction = record;
    Chisl: Integer;
     Znam; Word;
   End;
 Function nod(a,b:integer):integer;
  Var k:integer;
   Begin
     If a>b then k:= a else k:=b;
    While not((a mod k=0)and(b mod k =0)) do dec(k);
End;
   Var
     x,y,s: Tfraction
  n,p:integer;
 st: string;
begin
  writeln('Введите два рациональных числа');ъ
  write('x= '); readln(st); n:= post('/',st);
  val(copy(st,1,n-1),x.chisl,p);
  val(copy(st,n+1,length(st)-n),x.znam,p);
  write('y= '); readln(st); n:= post('/',st);
  val(copy(st,1,n-1),y.chisl,p);
  val(copy(st,n+1,length(st)-n),y.znam,p);
  {находим НОД для каждой дроби и сокращаем их}
  n:=nod(x.chisl,x.znam);
  x.chis:= x.chisl div n;
  x.znam:= x.znam div n;
  n:=nod(y.chisl,y.znam);
  x.chis:= y.chisl div n;
  x.znam:= y.znam div n;
 writeln('Сокращенные дроби:');
 writeln('X= ',x.chisl,'/',x.znam);
END.
```

Переведем на С++:

```
#include<stdio.h>
#include <iostream>
using namespace std;
long Nod(long a, long b)
{ while (a && b)
   if (a >= b) a %= b;
   else b %= a;
   return a | b;
}
```

```
int main( )
{
long a, b, nod;
cout<<"a="; cin>>a;
cout<<"b="; cin>>b;
nod  = Nod(a, b);
printf("%ld  %ld\n",  a/nod,      b/nod);
return 0;
}
```

Контрольные вопросы

1. Как описываются переменные типа запись в Pascal ?
2. В каких случаях целесообразно использовать переменные типа запись?Из каких компонентов состоит переменная типа запись?
3. Каков формат описания структурного типа на Си?
4. Какие операции допустимы над полями записи?
5. В чем отличие записей от массивов?

Задания для составления программ

Вариант 1	Багаж пассажира характеризуется количеством вещей и общим весом вещей. Сведения о багаже каждого пассажира представляют собой запись с двумя полями: одно поле целого типа (количество вещей) и одно - действительное (вес в килограммах). a) Найти багаж, средний вес одной вещи в котором отличается не более, чем на 0.3 кг от общего среднего веса одной вещи. b) Найти число пассажиров, имеющих более двух вещей и число пассажиров, количество вещей которых превосходит среднее число вещей. c) Определить, имеются ли два пассажира, багажи которых совпадают по числу вещей и различаются по весу не более чем на 0,5 кг. d) Выяснить, имеется ли пассажир, багаж которого превышает багаж каждого из остальных пассажиров и по числу вещей, и по весу.
Вариант 2	После поступления в ВУЗ о студентах собрана информация: фамилия, нуждается ли в общежитии, стаж, работал ли учителем, что окончил, какой язык изучал. Составить программу, определяющую: 1) сколько человек нуждаются в общежитии; 2) списки студентов, проработавших 2 и более лет учителем; 3) списки окончивших педучилище; 4) списки языковых групп.
Вариант 3	Описать, используя структуру данных запись, данные на студентов 1-го курса (фамилия, улица, дом, квартира). Составить программу, определяющую сколько студентов живет на улице Джамбула, списки студентов, живущих в доме номер 45.
Вариант 4	В библиотеке для каждого заказывающего книгу читателя заполняется карточка: фамилия, дата заказа, дата выдачи книги. Определить: 1) самый маленький срок, за который нашли книгу; 2) сколько заказов было не удовлетворено; 3) кто чаще всего берет книги; 4) кому выдали книги 15.09.90; 5) сколько человек заказывали книги 25.04.90.
Вариант 5	Описать, используя структуру данных запись, почтовую сортировку (город, улица, дом, квартира, кому, ценность). Составить программу, определяющую: 1) сколько посылок отправлено в г.Астану; 2) сколько и куда (список городов) отправлено посылок ценностью выше 10 рублей; 3) есть ли адреса куда отправлено более 1 посылки, если есть то сколько и кому.

Вариант 6	После поступления в ВУЗ о студентах собрана информация: фамилия, нуждается ли в общежитии, стаж, работал ли учителем, что окончил, какой язык изучал. Составить программу, определяющую: 1) сколько человек нуждаются в общежитии; 2) списки студентов, проработавших 2 и более лет учителем; 3) списки окончивших педучилище; 4) списки языковых групп.
Вариант 7	В школе было три 9 класса, в августе каждый классный руководитель имел сведения о своих учениках: фамилия, куда поступал, поступил или нет. Определить сколько учеников хотели пойти в 10 класс, кто хотел поступать в училище и техникум, кто поступил в училище или техникум, сколько учеников будет учится в 10 классе, сколько необходимо создать 10 классов и по сколько человек.
Вариант 8	На олимпиаде по информатике на школьников заполнялись анкеты: фамилия, номер школы, класс, занятое место. Напечатать: 1) списки школ, занявших призовые места; 2) какая из школ заняла больше всех призовых мест; 3) списки учеников занявших первое место, указать их класс.
Вариант 9	В деканате хранится информация о зимней сессии на 1 курсе (фамилия, номер группы, оценка 1 по геометрии, оценка 2 по алгебре, оценка 3 по информатике). Составить программу, печатающую фамилии студентов, имеющих задолженность хотя бы по одному предмету, качество успеваемости, процент студентов, т.е. сдавших экзамены на 4 и 5, название предмета, который был сдан лучше всего, номера групп в порядке убывания средней успеваемости их студентов.
Вариант 10	В отделе кадров студентов хранится следующая информация о каждом студенте: фамилия, имя, отчество, пол, возраст, курс. Составить программу которая печатает номер курса, на котором наибольший процент мужчин, самые распространенные мужские и женские имена, фамилии в алфавитном порядке и инициалы всех студенток, отчество и возраст которых являются одновременно самыми распространенными.

Литература основная

1. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
2. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
3. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.

Лабораторная работа 8
Программирование с использованием процедур и функций

Цель работы: закрепить понятия "процедура" и "функция" в языке программирования Pascal; понятие «функция» в C++, рассмотреть их сходства и различия, закрепить практические навыки работы с ЯП Pascal, C++ на примере реализации алгоритмов при помощи процедур и функций, научиться применять метод последовательной детализации в практическом программировании; применять процедуры и функции при решении задач.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Часто в программе обнаруживаются однотипные участки, которые выполняют одни и те же вычисления, но с различными данными. Такие части программы целесообразно оформлять в виде подпрограмм. В языке Pascal существует два вида подпрограмм: процедуры и функции, а в языке Си/C++ используется один тип подпрограмм-функция.

Перед выполнением данной работы необходимо изучить правила описания процедур и функций, механизм передачи параметров, ознакомиться с понятием локальной и глобальной переменной.

Каждая функция, вызываемая в программе, должна быть определена (только один раз). ***Определение функции*** – это ее полный текст, включающий заголовок и код.

Полное определение (реализация) функции на ЯП Си/C++имеет следующий вид:

тип_результата ID_функции(список параметров)
{
код функции
return выражение;
}

Для вызова функции в простейшем случае нужно указать ее имя, за которым в круглых скобках через запятую перечислить список передаваемых ей ***аргументов***. Вызов функции может находиться в любом месте программы, где по синтаксису допустимо выражение того типа, который формирует функция.

Простейший вызов функции имеет следующий формат:

ID_функции (***список аргументов***);

где в качестве аргументов можно использовать константы, переменные, выражения (их значения перед вызовом функции будут определены компилятором).

Аргументы в списке вызова должны совпадать со списком параметров вызываемой функции по количеству и порядку следования, а типы аргументов при передаче в функцию будут преобразованы, если это возможно, к типу соответствующих им параметров.

Глобальные переменные доступны всем функциям, где они не описаны как локальные переменные.

Функции могут располагаться в исходном файле в любом порядке, при этом исходная программа может размещаться в нескольких файлах.

Контрольные вопросы

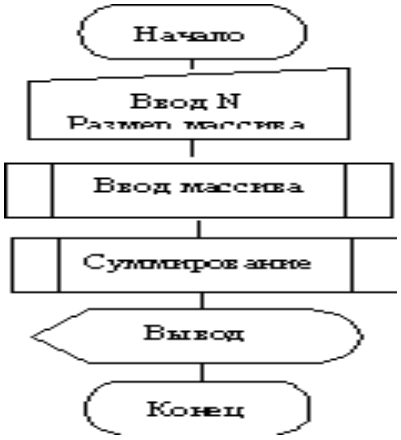
1. Для чего нужны в программе процедуры и функции?
2. В чем отличие между процедурой и функцией?
3. Чем отличаются формальные и фактические параметры?
4. Чем отличаются параметры-значения и параметры-переменные?
5. Как объявляются глобальные и локальные переменные? Каково правило видимости этих переменных?
6. Почему при обращении к процедуре, аргумент, передаваемый параметру-переменной, может быть только переменной, а не константой или выражением?

Пример. Найти сумму положительных элементов в массиве.

Этапы решения задачи:

1. Алгоритм решения довольно прост - в цикле будем "пробегать" массив, сравнивая его ячейки с 0 и суммировать, если они >0.
2. Составим блок-схему программы

Уточним из каких блоков состоит блок*"Суммирование положительных ячеек"*



Содержание этих блоков простое, поэтому не стоит их уточнять.

3. Напишем программу на языке Паскаль

```
program example;
  type
   Tarray = array[1..100] of integer;
    Var v: Tarray;
   N,i,s:integer;
  Procedure vvod_data(var m:Tarray;n:integer);
  Var i:integer;
Begin
  Writeln('Введите ',n,' чисел через пробел');
   For i:= 1 to n do read(m[i]);
End;
  Function summ(m:TArray):integer;
  Var s:integer;
Begin
   S:=0;
    For i:= 1 to n do if m[i]>0 then s:= s+m[i];
   Summ:=s;
End;
begin
     write('Введите размерность массива N= '); readln(n);
    vvod_Data(v,n);
   s:= summ(v);
  writeln('Сумма= ',s);
end.
```

Напишем программу на языке C++

```
#include <iostream>
#include <algorithm>
#include <numeric>
template <typename T>
struct PosSum {
   T operator () (const T & a, const T & b){
      return ( b > 0 ) ? a + b : a;
   }
};

int main(){
   const int SIZE = 5;
   int arr[SIZE] = { 1, -1, 2, -2, 0 };
```

```
    std::cout << "array: ";
    for ( int i = 0; i < SIZE; ++i )
        std::cout << arr[i] << ' ';
    std::cout << std::endl;
    std::cout << "Sum of positive elements: "
            << std::accumulate(arr, arr + SIZE, 0, PosSum<int>()) << std::endl;
    return 0;
}
```

Задания для составления программ

Вариант 1	Даны действительные числа x1, y1, x2, y2, …,x10, y10. Найти периметр десятиугольника, вершины которого имеют соответственно координаты (x1, y1), (x2, y2), …, (x10, y10). (Определить процедуру вычисления расстояния между двумя точками, заданными своими координатами.)
Вариант 2	Даны действительные числа a, b, c, d, e - стороны пятиугольника. Найти площадь пятиугольника. (Определить процедуру вычисления площади треугольника по его сторонам.)
Вариант 3	Даны три символьные матрицы. Найти a) ту матрицу, где есть хотя бы одна гласная - транспонировать; b) в той матрице, на главной диагонали которой все цифры, найти наименьшую и удалить соответствующую строку.
Вариант 4	Написать программу вычисления P(перестановок) по формуле:P=n!, где n - заданное натуральное число.
Вариант 5	Описать функцию Stepen (x,n) от вещественного x и целого n, вычисляющую (посредством умножения) величину xn, и использовать ее для вычисления b=2.7k+(a+1)-5.
Вариант 6	Даны отрезки a,b,c и d. Для каждой тройки этих отрезков, из которых можно построить треугольник, напечатать площадь данного треугольника. Определить процедуру Plos(x,y,z), печатающую площадь треугольника со сторонами x,y и z, если такой треугольник существует.
Вариант 7	Пусть процедура Socr(a,b,p,q) от целых параметров (b№0)приводит дробь $\frac{a}{b}$ к несократимому виду $\frac{p}{q}$ Описать данную процедуру и использовать ее для приведения дроби $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\Lambda+\frac{1}{20}$ к несократимому виду $\frac{c}{d}$
Вариант 8	Даны длины a,b и c сторон некоторого треугольника. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника. Длина медианы, проведенной к стороне a, равна $\frac{1}{2}\sqrt{2b^2+2c^2-a^2}$

Вариант 9	Даны координаты вершин двух треугольников. Определить, какой из них имеет большую площадь.
Вариант 10	Даны координаты вершин треугольника и координаты некоторой точки внутри него. Найти расстояние от данной точки до ближайшей стороны треугольника. (При определении расстояний учесть, что площадь треугольника вычисляется и через три его стороны, и через основание и высоту.).

Задачи повышенной сложности

1. Рассмотрим произвольное натуральное число и найдем сумму его цифр, затем сумму цифр полученного числа и так далее, пока не получим однозначное число. Назовем это число цифровым корнем. Требуется написать программу, которая для заданного N (N<10100) находит его цифровой корень.

2. Задано N натуральных чисел a1,a2,…,aN (1J N J 20), каждое из которых находится в интервале от 1 до 10000. Необходимо определить количество натуральных делителей произведения a1*a2*…*aN.

3. Написать функцию поиска корней полинома степени n. Исходными параметрами будут числа a1, a2,…,an. Комплексные корни учитывать.

4. Требуется написать программу, которая выводит в порядке возрастания все правильные несократимые дроби, знаменатели которых не превосходят N (2<=N<=500).

5. На экране компьютера, работающего в операционной системе Windows, было открыто N (N <= 20) окон, положение каждого из которых однозначно определяется четверкой натуральных чисел - X1 Y1 X2 Y2 - координатами левого верхнего и правого нижнего угла окна. Очевидно, что окна, открытые позже, могут частично или полностью перекрывать открытые ранее. Окно считается видимым, если виден хотя бы один образующий его пиксел.

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
3. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
4. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
5. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.

Лабораторная работа 9
Работа с файлами

Цель работы: закрепить навыки работы языке программирования Pascal с файловым типом данных (типизированные, текстовые и нетипизированные файлы); выработать навыки работы с файловым типом данных в ЯП С++, научиться считывать информацию из файлов, записывать информацию в файл; научится решать задачи с использованием текстовых и бинарных файлов.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения.

Файл – это набор данных, размещенный на внешнем носителе и рассматриваемый в процессе обработки как единое целое. В файлах размещаются данные, предназначенные для длительного хранения.

В языке Си не предусмотрены никакие заранее определенные структуры файлов. Все файлы рассматриваются компилятором как последовательность (поток байт) информации.

Для файлов определен маркер или указатель чтения-записи данных, который определяет текущую позицию доступа к файлу. Напомним, что с началом работы любой программы автоматически открываются стандартные потоки *stdin* и *stdout*.

В языке Си имеется большой набор функций для работы с файлами, большинство которых находятся в библиотеках ***stdio.h*** и ***io.h***. При этом потоки данных, с которыми работают функции ввода-вывода данных по умолчанию, буферизированы. Это означает, что при открытии потока с ним автоматически связывается определенный участок ОП, который и называется буфером. Все операции чтения-записи ведутся через этот буфер. Его размер фиксирован специальной константой *BUFSIZ*, которая определена в файле *stdio.h* как 512 (хотя программно ее можно изменять).

Файл представляет собой структурированный тип данных, содержащий последовательность компонентов одного типа и одной длины. Число элементов в файле (длина файла) не фиксировано. Это является основным отличием файла от массива.

Различают два вида файлов: *текстовые* и *бинарные*.

Текстовые файлы представляют собой последовательность ASCII символов и могут быть просмотрены и отредактированы с помощью любого текстового редактора. Эта последовательность символов разбивается на строки символов, при этом каждая строка заканчивается двумя кодами «перевод строки», «возврат каретки»: 13 и 10 (0xD и 0xA).

Бинарные (двоичные) файлы представляют собой последовательность данных, структура которых определяется программно.

Прежде чем начать работать с файлом, т.е. получить возможность чтения или записи информации в файл, его нужно открыть для доступа.

Для этого обычно используется функция

FILE* ***fopen***(char * ID_файла, char *режим);

w – файл открывается для записи (*write*); если файла с заданным именем нет, то он будет создан; если же такой файл уже существует, то перед открытием прежняя информация уничтожается;

r – файл открывается для чтения (*read*); если такого файла нет, то возникает ошибка;

a – файл открывается для добавления (*append*) новой информации в конец;

r+ (*w+*) – файл открывается для редактирования данных, т.е. возможны и запись, и чтение информации;

a+ – то же, что и для *a*, только запись можно выполнять в любое место файла (доступно и чтение файла);

t – файл открывается в текстовом режиме;

b – файл открывается в двоичном режиме;

Последние два режима используются совместно с рассмотренными выше. Возможны следующие комбинации режимов доступа: “*w+b*”, “*wb+*”, “*rw+*”, “*w+t*”, “*rt+*”, а также некоторые другие комбинации.

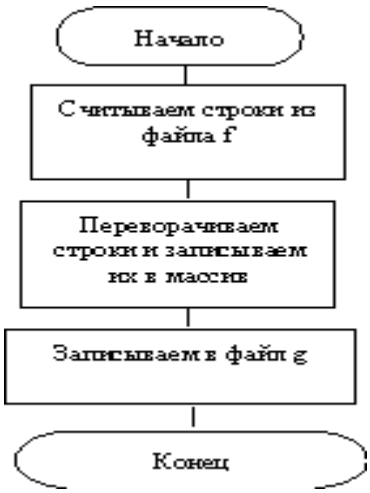
По умолчанию файл открывается в текстовом режиме.

Текстовый режим отличается от двоичного тем, что при открытии файла как текстового пара символов «перевод строки» и «возврат каретки» заменяется на один символ «перевод строки» для всех функций записи данных в файл, а для всех функций вывода – наоборот – символ «перевод строки» заменяется на два символа – «перевод строки» и «возврат каретки».

Пример. Переписать из текстового файла f в файл g строки в перевернутом виде, порядок строк должен быть обратным.

Этапы решения задачи:

1. Будем считывать файл f построчно и перевертывая строки будем записывать их в массив, далее создадим файл и заполним его строками из массива меняя порядок строк на обратный.
2. Составим блок -схему программы.



3.Реализации каждого блока на Паскале.

а) "Считываем строки из файла F"

```
assign(f,'input.dat');
 reset(f);
 while not eof(f) do
begin readln(f,st);
end;
```

б) "Переворачиваем строки и записываем их в массив"

```
st1:='';
 for i:= 1 to length(st) do
  st1:= st[i]+st1;
 m[k]:=st1;
k:=k+1;
```

в) "Записываем в файл g"

```
assign(g,'output.dat');
rewrite(g);
for i:= 1 to k do writeln(g,m[i]);
```

4. Программа целиком

```
program example;
    var f,g:text; k,i:integer;
    m:array[1..100] of string;
begin
    assign(f,'input.dat');
    reset(f); k:=0;
    while not eof(f) do
     begin readln(f,st);
       k:=k+1; st1:='';
       for i:= 1 to length(st) do
    st1:= st[i]+st1;
          m[k]:=st1;
       end;
     assign(g,'output.dat');
     rewrite(g);
     for i:= 1 to k do
writeln(g,m[i]);
    end.
```

Программа на ЯП C++

```
#include <fstream>
using namespace std;
 int main()
{
   ifstream in("in.txt");
   ofstream out("out.txt");
   char c;
for(in.seekg(-1, ios::end);
in;cin.peek(),out<<cin.seekg(-1, ios::cur));
out.close(), in.close();
}
```

Контрольные вопросы

1. Что такое файл? Какие существуют виды файлов?
2. Какими стандартными процедурами и функциями располагает Pascal, C++ для работы с файлами?
3. Каково должно быть содержание программы по созданию файла и его корректировки (замена элементов, добавление элементов, удаление элементов)?
4. Каковы особенности работы с текстовыми файлами?
5. Каковы особенности работы с типизированными файлами?

Задания для составления программ :

Вариант 1	Даны текстовые файлы f1 и f2. Переписать с сохранением порядка следования компоненты файла f1 в файл f2, а компоненты файла f2 в файл f1. Использовать вспомогательный файл h.
Вариант 2	Дан текстовый файл f. Записать в файл g компоненты файла f в обратном порядке.
Вариант 3	Даны текстовые файлы f и g. Записать в файл h сначала компоненты файла f, затем - компоненты файла g с сохранением порядка.
Вариант 4	Дан файл f, компоненты которого являются целыми числами. Получить в файле g все компоненты файла f: а) являющимися четными числами; б) делящиеся на 3 и не делящиеся на 7; в) являющимися точными квадратами.
Вариант 5	Дан файл f, компоненты которого являются целыми числами. Получить файл g, образованный из файла f исключением повторных вхождений одного и того же числа.
Вариант 6	Дан файл f, компоненты которого являются целыми числами. Никакая из компонент файла не равна нулю. Файл f содержит столько же отрицательных чисел, сколько и положительных. Используя вспомогательный файл h, переписать компоненты файла f в файл g так, чтобы в файле g: а) не было двух соседних чисел с одинаковым знаком; b) вначале шли положительные, затем отрицательные числа;
Вариант 7	Дан файл f, компоненты которого являются целыми числами. Никакая из компонент файла не равна нулю. Числа в файле идут в следующем порядке: десять положительных, десять отрицательных, десять положительных, десять отрицательных и т.д. Переписать компоненты файла f в файл g так, чтобы в файле g числа шли в следующем порядке: а) пять положительных, пять отрицательных, пять положительных, пять отрицательных и т.д.; б) двадцать положительных, двадцать

	отрицательных, двадцать положительных, двадцать отрицательных и т.д. (предполагается, что число компонент в файле f делится на 40).
Вариант 8	Дан файл f, компоненты которого являются целыми числами. Записать в файл g наибольшее значение первых пяти компонент файла f, затем - следующих пяти компонент и т.д. Если в последней группе окажется менее пяти компонент, то последняя компонента файла g должна быть равна наибольшей из компонент файла f, образующих последнюю (неполную) группу.
Вариант 9	Дан символьный файл f: а) подсчитать число вхождений в файл сочетаний 'ab'; б) определить входит ли в файл сочетание 'abcdefgh'; в) подсчитать число вхождений в файл каждой из букв 'a','b','c','d', 'e','f' и вывести результат в виде таблицы a --> Na b --> Nb c --> Nc d --> Nd e --> Ne f --> Nf где Na, Nb, Nc, Nd, Ne, Nf - числа вхождений соответствующих букв.
Вариант 10	Сведения об автомобиле состоят из его марки, номера и фамилии владельца. Дан файл f, содержащий сведения о нескольких автомобилях. Найти: фамилии владельцев и номера автомобилей данной марки; Найденные данные записать в файл g.
Вариант 11	Сведения об ученике состоят из его имени и фамилии и названия класса (года обучения и буквы), в котором он учится. Дан файл f, содержащий сведения об учениках школы: а) выяснить, имеются ли в школе однофамильцы; в) выяснить, имеются ли однофамильцы в каких-либо параллельных классах; с) выяснить, в каких классах насчитывается более 35 учащихся;
Вариант 12	Дан файл f, содержащий различные даты. Каждая дата - это число, месяц и год. Найти: а) год с наименьшим номером; б) все весенние даты; в) самую позднюю дату. Найденные данные записать в файл g.
Вариант 13	Дан файл f, содержащий сведения о книгах. Сведения о каждой из книг - это фамилия автора, название и год издания. 1) Найти названия книг данного автора, изданных с 1960 г. 2) Определить, имеется ли книга с названием "Информатика". Если да, то сообщить фамилию автора и год издания. Если таких книг несколько, то сообщить имеющиеся сведения обо всех книгах.
Вариант 14	Дан файл f, содержащий сведения о кубиках: размер каждого кубика (длина ребра в сантиметрах), его цвет (красный, зеленый, желтый или синий) и материал

	(деревянный, металлический, картонный). Найти: а) количество кубиков каждого из перечисленных цветов и их суммарный объем; б) количество деревянных кубиков с ребром 3 см и количество металлических кубиков с ребром, большим 5 см.
Вариант 15	Дан файл f, содержащий сведения о веществах: указывается название вещества, его удельный вес и проводимость (проводник, полупроводник, изолятор). 1) Найти удельные веса и названия всех полупроводников. 2) Выбрать данные о проводниках и упорядочить их по убыванию удельных весов.

Задачи повышенной сложности

1. Написать программу для сжатия и распаковки фалов: а) текстового, б) типизированного (file of string), в) нетипизированного.
2. Дан текстовый фал, содержащий программу на языке Паскаль. Проверить эту программу на соответствие числа открывающихся и закрывающихся скобок (любых).
3. Дан нетипизированный файл. Записать файл "в обратном порядке" (с изменением порядка байтов).

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
3. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
4. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
5. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
6. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.
7. Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.

Лабораторная работа №10
Сортировка методом простого выбора

Цель работы: Исследовать сортировку массива методом простого выбора и оценить эффективность этого алгоритма.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

При **сортировке массива** методом простого выбора применяется базовый алгоритм поиска максимального (минимального) элемента и его номера.

Алгоритм сортировки массива методом выбора:

1. Для исходного массива выбрать максимальный элемент.
2. Поменять его местами с последним элементом (после этого самый большой элемент будет стоять на своем месте).
3. Повторить п.п. 1-2 с оставшимися n-1 элементами, то есть рассмотреть часть массива, начиная с первого элемента до предпоследнего, найти в нем максимальный элемент и поменять его местамис предпоследним (n-1)- м элементом массива, затем с оставшиеся (n-2)-мя элементами и так далее, пока не останется один элемент, уже стоящий на своем месте.

Для упорядочения массива потребуется (n-1) просмотров массива. В процессе сортировки будет увеличиваться отсортированная часть массива, а неотсортированная, соответственно, уменьшаться.

При сортировке данных выполняется обмен содержимого переменных. Для обмена необходимо создавать временную переменную, в которой будет храниться содержимое одной из переменных. В противном случае ее содержимое окажется утерянным.

Пример

Опишем процедуру сортировки на языке проектирования программ (псевдокоде).

```
For i := n downto 2 do
  Begin
    найти максимальный элемент из a[1], ..., a[i];
     запомнить его индекс в переменной k;
    если i<>k поменять местами a[i] и a[k];
End;
```

Вот как изменяется значение массива из пяти элементов (30,20,10,50,40)

30	20	10	50	40
30	20	10	40	50
30	20	10	40	50
10	20	30	40	50
10	20	30	40	50

Подчеркнута область поиска наибольшего элемента.

Реализация алгоритма на С++

```
#include <iostream>
using namespace std;
int main ()
{
   const int n=5;
   int b[n];
   int i, a, j;
   for (i=0; i<n; i++)
   {
      cin>>b[i];
   }
   for (i=0; i<n-1; i++)
   {
      for (j=i+1; j<n; j++)
      {
         if(b[j]<b[i]){
         a=b[i];
         b[i]=b[j];
         b[j]=a;
         }
      }
   }
   for (i=0; i<n; i++)
   {
      cout<<b[i]<<" ";
   }
   return 0;
}
```

Контрольные вопросы

1. Что понимается под сортировкой массива?
2. Чем отличается сортировка по убыванию от сортировки по возрастанию?
3. Сформулировать идею сортировки массива методом простого выбора.
4. Почему время выполнения одного шага прямо пропорционально размеру неупорядоченной части массива?

Варианты заданий для составления программ

Входные данные (исходный массив) и выходные данные (отсортированный массив) формировать в виде текстового файла, содержащего целые числа!

Вариант 1	Изменить процедуру сортировки так, чтобы сортировка производилась по убыванию элементов.
Вариант 2	Проверить, является ли данная последовательность целых чисел упорядоченной по убыванию. Если нет, упорядочить ее.

Вариант 3	Отсортировать данный массив и подсчитать количество уникальных чисел в массиве.
Вариант 4	Изменить процедуру сортировки так, чтобы значение параметра i с каждым шагом увеличивалось.
Вариант 5	Отсортировать четные элементы массива с помощью простого выбора.
Вариант 6	Отсортировать с помощью простого выбора элементы массива, стоящие на нечетных местах.
Вариант 7	Отсортировать положительные элементы массива с помощью простого выбора.
Вариант 8	Отсортировать отрицательные элементы массива с помощью простого выбора.
Вариант 9	В матрице n*m отсортируйте столбцы в порядке возрастания.
Вариант 10	Даны список футбольных команд высшей лиги и количество очков, набранных каждой командой в чемпионате. Известно, что нет команд с равным числом очков. Распечатать список призеров.
Вариант 11	В неупорядоченном массиве могут быть совпадающие элементы. Из каждой группы одинаковых элементов оставить только один, удалив остальные и «поджав» массив к его началу.
Вариант 12	Турнирная таблица соревнований представлена квадратной матрицей A, каждый элемент которой aij есть число голов, забитых i-ой командой в ворота j-ой команды. По диагонали расположить место каждой команды (по числу побед за вычетом числа поражений; в случае равенства – по разности забитых и пропущенных голов).

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Немнюгин С.А. Turbo PASCAL:Практикум:Учебное пособие/ МОРФ.-2 изд.-М.:СПб.,2005.-267 с.
3. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
4. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
5. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
6. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.
7. Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.

Лабораторная работа №11
Сортировка методом простого обмена

Цель работы:

Исследовать сортировку массива методом простого обмена и оценить эффективность этого алгоритма.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++,стандартные библиотеки используемых языков программирования

Общие сведения.

Алгоритм сортировки методом простого обмена состоит из отдельных шагов:

- Слева направо поочередно сравниваются два соседних элемента, и если их взаимное расположение не соответствует заданному условию упорядоченности, то они меняются местами. Далее берутся два следующих соседних элемента и так далее до конца массива.
- После одного такого прохода на последней n-ой позиции массива будет стоять максимальный (или минимальный) элемент ("всплыл" первый "пузырек"). Поскольку максимальный (или минимальный) элемент уже стоит на своей последней позиции, то второй проход обменов выполняется до (n-1)-го элемента. И так далее. Всего требуется (n-1) проход.
- На каждом шаге проходят массив от начала к концу, сравнивая пары соседних элементов. Если очередная пара нарушает требуемый порядок, ее элементы меняют местами. Шаги повторяют до тех пор, пока очередной проход не вызовет ни одного обмена.

Пример. Отсортируем по возрастанию методом простого обмена массив из 5 элементов: 5 1 8 4 9. Длина текущей части массива – n-k+i, где k – номер просмотра, i – номер проверяемой пары, n - k – номер последней пары. За вертикальной чертой располагаются отсортированные элементы.

Первый просмотр: рассматривается весь массив.

i = l5 4 8 2 9
> меняем
i = 24 5 8 2 9
< не меняем
i = 34 5 8 2 9
> меняем
i = 44 5 2 8 9
< не меняем

9 стоит на своем месте.

Второй просмотр: рассматриваем часть массива с первого до четвертого элемента.

i = 14 5 2 8 9
< не меняем
i = 24 5 2 8 9
> меняем
i = 34 2 5 8 9
< не меняем

8 стоит на своем месте.

Третий просмотр: рассматриваемая часть массива содержит первые три элемента.

i = 14 2 5 8 9
> меняем
i = 22 4 5 8 9
< не меняем

5 стоит на своем месте.

Четвертый просмотр: рассматриваем последнюю пару.

i = 12 4 5 8 9
< не меняем

4 стоит на своем месте.

Для самого маленького элемента (2) остается только одно место – первое.

Итак, наш массив отсортирован по возрастанию элементов методом простого обмена. Этот метод также называют методом «пузырька». Название это происходит от образной интерпретации, при которой в процессе выполнения сортировки более «легкие» элементы (элементы с заданным свойством) мало-помалу всплывают на «поверхность».

Реализация на C++

```
#include <iostream>
using namespace std;
void bubbleSort(int* arr, int size)
{   int tmp, i, j;
    for(i = 0; i < size - 1; ++i)
    {  for(j = 0; j < size - 1; ++j)
       {
          if (arr[j + 1] < arr[j])
          {
             tmp = arr[j + 1];
             arr[j + 1] = arr[j];
             arr[j] = tmp;
          }
       }
    }
}
```

Контрольные вопросы

1. Как поменять местами два элемента массива?
2. Что такое вложенные циклы?
3. В чем сходство и отличие методов простого выбора и простого обмена?

ВНИМАНИЕ!!! Входные данные (исходный массив) и выходные данные (отсортированный массив) формировать в виде текстового файла, содержащего целые числа! Для всех вариантов предварительно написать процедуру «пузырьковой» сортировки.

Варианты заданий для составления программ:

Вариант 1	Подсчитать количество выполненных сравнений и перестановок в лучшем, худшем случаях и в среднем.
Вариант 2	Упорядочить первые n элементов данного ряда в порядке возрастания. Напечатать эти элементы в порядке убывания.
Вариант 3	В нашем примере два последних прохода не влияют на порядок элементов, так как они уже отсортированы. Следовательно, можно улучшить наш алгоритм, если запоминать, производились ли перестановки элементов в процессе некоторого прохода. Если их не было, то сортировку можно закончить. Модифицировать процедуру с учетом этой возможности улучшения.
Вариант 4	Если известен не только факт последнего обмена, но и его место, то нетрудно заметить, что все пары соседних элементов, расположенные правее этого места, уже находятся в нужном порядке. Поэтому просмотр можно закончить на этом индексе, а не продолжать до конца. Модифицировать процедуру с учетом этой возможности улучшения.
Вариант 5	Написать процедуру «пузырьковой» сортировки по убыванию.
Вариант 6	Найти второй по величине (после наименьшего) элемент данного ряда.
Вариант 7	Найти так называемую медиану ряда, т. е. такой его элемент, который больше любого из одной половины элементов и меньше любого из другой (если число элементов ряда четно, следует взять среднее значение из двух значений, обладающих указанным свойством).
Вариант 8	В методе «пузырьковой» сортировки если какие-то два соседние элементы ряда (например, a[4] и a[5]) не упорядочены, они переставляются, после чего алгоритм предписывает перейти к сравнению следующей пары – a[5] и a[6]. Вместо этого можно попытаться проследить за элементом a[4], дав ему возможность «всплывать» в сторону меньших индексных позиций ряда. Для этого сравним a[4] с a[3]. Если окажется, что a[3] больше,

	осуществим перестановку и затем сравним a[3] с a[2] и т. д. Осуществить указанный метод.
Вариант 9	В целочисленном массиве найти наибольшее число одинаковых элементов.
Вариант 10	Дано n целых чисел. Сколько чисел лежит между данными a и b?
Вариант 11	Дано n отрезков [a[i], b[i]] на прямой (i=1..n). Найти максимальное k, для которого существует точка прямой, покрытая k отрезками ("максимальное число слоев"). Подсказка. Упорядочим все левые и правые концы отрезков вместе (при этом левый конец считается меньше правого конца, расположенного в той же точке прямой). Далее двигаемся слева направо, считая число слоев. Встреченный левый конец увеличивает число слоев на 1, правый - уменьшает.
Вариант 12	Дано n точек на плоскости. Указать (n-1)-звенную несамопересекающуюся незамкнутую ломаную, проходящую через все эти точки. (Соседним отрезкам ломаной разрешается лежать на одной прямой.) Подсказка. Упорядочим точки по x-координате, а при равных x-координатах - по y-координате.

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
3. Немнюгин С. А. Turbo Pascal;практикум./МОРФ.-СПб:Питер,2003.-256с.:ил.
4. Немнюгин С.А. Turbo PASCAL:Практикум:Учебное пособие/ МОРФ.-2 изд.-М.:СПб.,2005.-267 с.
5. Культин Н. Самоучитель:Программирование в Turbo Pascal7.0 и Delphi.-2 изд.- Спб.:БХВ-Петербург,2003.-404 с
6. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
7. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
8. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
9. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.
10. Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.

Лабораторная работа №12
Сортировка методом прямого включения

Цель работы: Исследовать сортировку массива методом прямого включения и оценить эффективность этого алгоритма.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Сортировка методом прямого включения, так же, как и сортировка методом простого выбора, обычно применяется для массивов, не содержащих повторяющихся элементов.

Сортировка этим методом производится последовательно шаг за шагом. На k-м шаге считается, что часть массива, содержащая первые k-1 элемент уже упорядочена, то есть $a[1] \le a[2] \le \ldots \le a[k-1]$. Далее необходимо взять k-й элемент и подобрать для него место в отсортированной части массива такое, чтобы после его вставки упорядоченность не нарушилась, то есть надо найти такое $j\,(1 \le j \le k-1)$ что $a[j] \le a[k] \le a[j+1]$. Затем надо вставить элемент a[k] на найденное место.

С каждым шагом отсортированная часть массива увеличивается. Для выполнения полной сортировки потребуется выполнить n-1 шаг.

Осталось ответить на вопрос, как осуществить поиск подходящего места для элемента x. Поступим следующим образом: будем просматривать элементы, расположенные левее x (то есть те, которые уже упорядочены), двигаясь к началу массива. Нужно просматривать элементы a[j], j изменяется от k-l до 1. Такой просмотр закончится при выполнении одного из следующих условий:

1) найден элемент a[j]<x, что говорит о необходимости вставки x между a[j-1] и a[j];

2) достигнут левый конец упорядоченной части массива, следовательно, нужно вставить x на первое место.

До тех пор, пока одно из этих условий не выполнится, будем смещать просматриваемые элементы на 1 позицию вправо, в результате чего в отсортированной части будет освобождено место под x.

Пример

Коротко опишем фрагмент алгоритма сортировки с помощью прямого включения:

```
For k := 2 to n do
  begin
    x := a[k]; j := k-1;
    { вставить x на подходящее место в a[1], …, a[k] }
    { для этого организуем цикл, которые выполняется, пока }
    { j > 0 и x <= a[j] }
```

```
    {в цикле смещаем элементы массива на 1 позицию вправо }
    {по выходу из цикла вставляем x в позицию j+1массива }
  end;
```

Программа на C++

```
#include <iostream>
#include <cstdlib>
#include <ctime>
 using namespace std;
 int main() {
    srand(time(NULL));
    int a[7], tmp, j, i;
     for (i = 0; i<7; i++)
        a[i] = rand()%20;
    for (i = 0; i<7; i++)
        cout << a[i] << " ";
    cout << endl;
    for (i = 0; i<7; i++) {
        tmp = a[i];
        j = i-1;
        while ((j>=0) && (a[j]>tmp)) {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = tmp;
    }
    for (i = 0; i<7; i++)
        cout << a[i] << " ";
    return 0;
}
```

Контрольное задание

Написать программу вставки последнего элемента массива после первого отрицательного элемента этого же массива.

Варианты заданий

Если явно не указано иное, входные данные (исходный массив) и выходные данные (отсортированный массив) формировать в виде текстового файла, содержащего целые числа!

Для всех заданий предварительно написать процедуру сортировки массива методом прямого включения

Вариант 1	Дан ряд, содержащий n элементов. Отсортировать их в порядке возрастания, отбрасывая при этом все повторяющиеся элементы.

Вариант 2	Определить моду данного ряда – значение, встречающееся среди его элементов чаще всего.
Вариант 3	Исходный набор данных представляет собой последовательность записей, состоящих из фамилии, возраста и стажа работы. Распечатать этот список: 1) в алфавитном порядке; 2) в порядке увеличения возраста; 3) в порядке увеличения стажа работы.
Вариант 4	Написать процедуру сортировки по убыванию.
Вариант 5	Дан ряд целых чисел. Получить в порядке возрастания все различные числа, входящие в этот ряд.
Вариант 6	Дан ряд из n различных целых чисел. Получить различные целые числа $i_1, i_2, ..., i_n$ такие, что $a_{i_1} < a_{i_2} < ... < a_{i_n}$.
Вариант 7	Даны целые $a_1, a_2, ..., a_n$. Найти наибольшее значение в этой последовательности после выбрасывания из нее всех членов со значением $\max(a_1, a_2, ..., a_n)$
Вариант 8	Даны натуральные $n, a_1, ..., a_n$ $(n \geq 4)$. Числа $a_1, ..., a_n$ – это измеренные в сотых долях секунды результаты n спортсменов в беге на 100 м. Составить команду из четырех лучших бегунов для участия в эстафете 4x100, т. е. указать одну из четверок натуральных чисел i, j, k, l такую, что сумма $a_i + a_j + a_k + a_l$ имеет наименьшее значение.
Вариант 9	Дано n независимых случайных точек, с координатами (xi; yi), равномерно распределенных в круге радиуса 1 с центром в начале координат. Напишите программу, располагающую точки в порядке возрастания расстояния от центра.
Вариант 10	Даны n целых положительных двузначных чисел. Трактуя каждое число как пару цифр из интервала 0–9, отсортировать их (цифры) по возрастанию.

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
3. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
4. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
5. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.
6. Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.

Лабораторная работа №13
Рекурсия

Цель работы: закрепить один из эффективных способов решения сложных задач – с использованием рекурсии в Pascal, рекурсивных функций в Си/C++

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения: В математике рекурсивным называется определение любого понятия через самое себя. В программировании часто, разрабатывая программу, удается свести исходную задачу к более простой. Среди этих задач может оказаться и первоначальная, но в упрощенной форме..

Алгоритм называется рекурсивным, если он прямо или косвенно обращается к самому себе. Часто в основе такого алгоритма лежит рекурсивное определение какого-то понятия. Например, о факториале числа N можно сказать, что N! = N*(N – 1)!, если N > 0 и N! = 1 если N = 0. Это – рекурсивное определение. Иными словами, частью алгоритма вычисления функции будет вычисление этой же функции.

Рекурсивной (самовызываемой или самовызывающей) называют функцию, которая прямо или косвенно вызывает сама себя.

При каждом обращении к рекурсивной функции создается новый набор объектов автоматической памяти, локализованных в коде функции.

Возможность прямого или косвенного вызова позволяет различать прямую или косвенную рекурсии. Функция называется косвенно рекурсивной в том случае, если она содержит обращение к другой функции, содержащей прямой или косвенный вызов первой функции. В этом случае по тексту определения функции ее рекурсивность (косвенная) может быть не видна. Если в функции используется вызов этой же функции, то имеет место прямая рекурсия, т.е. функция по определению рекурсивная.

Рекурсивные алгоритмы эффективны в задачах, где рекурсия использована в самом определении обрабатываемых данных. Поэтому изучение рекурсивных методов нужно проводить, вводя динамические структуры данных с рекурсивной структурой. Рассмотрим вначале только принципиальные возможности, которые предоставляет язык Си для организации рекурсивных алгоритмов.

В рекурсивных функциях необходимо выполнять следующие правила.

1. При каждом вызове в функцию передавать модифицированные данные.

2. На каком-то шаге должен быть прекращен дальнейший вызов этой функции, это значит, что рекурсивный процесс должен шаг за шагом упрощать задачу так, чтобы для нее появилось нерекурсивное решение, иначе функция будет вызывать себя бесконечно.

3. После завершения очередного обращения к рекурсивной функции в вызывающую функцию должен возвращаться некоторый результат для дальнейшего его использования.

***Пример* 1**. Заданы два числа *a* и *b*, большее из них разделить на меньшее, используя рекурсию.

Текст программы может быть следующим:

```
double proc(double, double);
void main (void)
{
        double a,b;
        puts(“ Введи значения a, b : ”);
        scanf(“%lf %lf”, &a, &b);
        printf(“\n Результат деления : %lf”, proc(a,b));
}
C++
#include <iostream>
#include <cstring>
using namespace std;
 int main ()
{
   int a,b;
        cout<<"a="; cin>>a;
   cout<<"b="; cin>>b;

        {
              if (b == 0)
                 return a;
              else
                 return (b, a % b);
        }
}
```

Любое рекурсивное определение состоит из двух частей. Эти части принято называть базовой и рекурсивной частями. Базовая часть является нерекурсивной и задает определение для некоторой фиксированной части объектов. Рекурсивная часть определяет понятие через него же и записывается так, чтобы при цепочке повторных применений она редуцировалась бы к базе.

Пример2

Задача. Написать рекурсивную программу поиска минимального элемента массива.

Решение. Опишем функцию Pmin, которая определяет минимум среди первых n элементов массива a. Параметрами этой функции являются количество элементов в рассматриваемой части массива - n и значение последнего элемента этой части – a[n]. При этом если n>2, то результатом является минимальное из двух чисел – a[n] и минимального числа из первых (n-1) элементов массива. В этом заключается рекурсивный вызов. Если же n=2, то результатом является минимальное из первых двух элементов массива. Чтобы найти минимум всех элементов массива, нужно обратиться к функции Pmin, указав в качестве параметров значение размерности массива и значение последнего его элемента. Минимальное из двух чисел определяется с помощью функции Min, параметрами которой являются эти числа.

```
Program Example _1;
Const n=10;
Type MyArray=Array[1..n] of Integer;
Const a : MyArray = (4,2, -1,5,2,9,4,8,5,3);
Function Min (a, b : Integer) : Integer;
      Begin
        if a>b then Min := b else Min:=a;
      End;
Function Pmin(n, b : Integer) : Integer;
      Begin
        if n = 2 then Pmin := Min(n,a[1]) else
        Pmin := Min(a[n], Pmin(n-1,a[n]));
      End;
BEGIN
  Writeln('Минимальный элемент массива  - ', Pmin(n,a[n]));
END.
```

Программа на C++:

```
#include<iostream>
using namespace std;

int is_center(int left, int right)
{
  return !((left + right) % 2);
}

int min(const int a[], int left, int  right)
{
  int x, y, m, center;
  if (left == right) return a[left];
  m = (left + right) / 2;
  if(is_center(left, m))
    x = min(a, left, m);
```

```
  else
    x = min(a, left, m - 1);
  if(is_center(m, right))
    y = min(a, m, right);
  else
    y = min(a, m + 1, right);
  if (x < y) return x;
  else return y;
}
int main()
{
  int mas[5]={5,4,10,2,6};
  cout << min(mas, 0, 4) << endl;
}
```

3. Задача. Ханойские башни. Имеется три стержня A, B, C. На стержень A нанизано n дисков радиуса 1, 2,..., n таким образом, что диск радиуса i является i-м сверху. Требуется переместить все диски на стержень B, сохраняя их порядок расположения (диск с большим радиусом находится ниже). За один раз можно перемещать только один диск с любого стержня на любой другой стержень. При этом должно выполняться следующее условие: на каждом стержне ни в какой момент времени никакой диск не может находиться выше диска с меньшим радиусом.

Решение. Предположим, что мы умеем перекладывать пирамиду из (n-1) диска. Рассмотрим пирамиду из n дисков. Переместим первые (n-1) дисков на стержень C (это мы умеем). Затем перенесем последний n-й диск со стержня A на стержень B. Далее перенесем пирамиду из (n-1) диска со стержня C на стержень B. Так как n-й диск самый большой, то условие задачи не будет нарушено. Таким образом, вся пирамида будет на стержне B. Аналогичным образом можно перенести n – 2, n – 3 и т. д. дисков. Когда n=1, осуществить перенос очень просто: непосредственно с первого стержня на второй. При этом для решения задачи будет достаточно 2n - 1 перекладываний.

```
Program Example_2;
Const k = 3;
Var a,b,c : Char;
Procedure Disk(n : Integer; a, b, c: Char);
Begin
  if n>0 then
     begin
       Disk(n-1,a,c,b);
       Writeln('Диск ',n, ' с ', a,'->', b);
       Disk(n-1,c,b,a);
     end;
```

```
End;
BEGIN
  a := ‘A’; b := ‘B’; c := ‘C’;
  Disk(k,a,b,c);
 ReadLn;
END.
```

Программа задачи на C++:

```
#include <iostream>
using namespace std;
void hanoi_towers(int quantity, int from, int to, int buf_peg)
{
	if (quantity != 0)
	{
		hanoi_towers(quantity-1, from, buf_peg, to);
		cout << from << " -> " << to << endl;
		hanoi_towers(quantity-1, buf_peg, to, from);
	}
}
 int main()
{	setlocale(LC_ALL,"rus");
	int start_peg, destination_peg, buffer_peg, plate_quantity;
	cout << " Номер первого столбика::" << endl;
	cin  >> start_peg;
	cout << " Номер конечного столбика:" << endl;
	cin  >> destination_peg;
	cout << " Номер промежуточного столбика:" << endl;
	cin  >> buffer_peg;
	cout << " Количество дисков:" << endl;
	cin  >> plate_quantity;
	 hanoi_towers(plate_quantity, start_peg, destination_peg, buffer_peg);
return 0;
}
```

Контрольные вопросы

1. На чем основан рекурсивный метод программирования?
2. В чем разница между «циклическим» и «рекурсивным» способами определения? Какой элемент является обязательным в рекурсивном определении?
3. Что такое «фрейм активации»?
4. К каким последствиям приводит «рекурсивное зацикливание»?
5. Какое условие должно обязательно присутствовать в любой рекурсивной процедуре?
6. Что такое явная и косвенная рекурсии?
7. Дайте рекурсивное определение целой степени числа N.

Варианты заданий для составления программ

Вариант 1	Ввести последовательность чисел (окончание ввода – 0) и вывести их в обратной последовательности. Входные данные взять из текстового файла.
Вариант 2	Используя команды write(x) лишь при x=0..9, написать рекурсивную программу печати десятичной записи целого положительного числа n.
Вариант 3	Напишите рекурсивную функцию, которая возвращает среднее из n элементов массива чисел.
Вариант 4	Найти первые N чисел Фибоначчи двумя способами: с помощью рекурсии и с помощью итерации. Сравнить эффективность алгоритмов.
Вариант 5	Написать функцию сложения двух чисел, используя только прибавление единицы.
Вариант 6	Написать функцию умножения двух чисел, используя только операцию сложения.
Вариант 7	Вычислить сумму элементов одномерного массива.
Вариант 8	Найти НОД (наибольший общий делитель) двух натуральных чисел.
Вариант 9	Вычислить несколько значений функции Аккермана для неотрицательных чисел m и n: $A(n,m)=\begin{cases} m+1, & n=0 \\ A(n-1,1), & n\neq 0, m=0 \\ A(n-1,A(n,m-1)), & n>0, m\geq 0 \end{cases}$
Вариант 10	Напишите рекурсивную функцию, которая вычисляет длину строки

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
3. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
4. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.

Лабораторная работа №14
Линейные списки

Цель работы: Получить практические навыки работы с динамическими переменными и динамическими структурами данных.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения

Для работы с динамическими структурами данных используются указатели. Указатели представляют собой специальный тип данных. Они принимают значения, равные адресам размещения в оперативной памяти соответствующих динамических переменных.

Списком называется структура данных, каждый элемент которой посредством указателя связывается со следующим элементом. На самый первый элемент (голову списка) имеется отдельный указатель.

Из определения следует, что каждый элемент списка содержит поле данных (оно может иметь сложную структуру) и поле ссылки на следующий элемент. После ссылки последнего элемента должно содержать пустой указатель.

Число элементов связанного списка может расти или уменьшаться в зависимости от того, сколько данных мы хотим хранить в нем. Чтобы добавить новый элемент в список, необходимо:

1. Получить память для него;
2. Поместить туда информацию;
3. Добавить элемент в конец списка (или начало).

Элемент списка состоит из разнотипных частей (хранимая информация и указатель), и его естественно представить записью. Перед описанием самой записи описывают указатель на нее:

```
Type { описание списка из целых чисел }
  PList = ^TList;
  TList = record
     Inf : Integer;
     Next : PList;
  end;
```

Примеры

Создание списка.

Задача. Сформировать список, содержащий целые числа 3, 5, 1, 9.

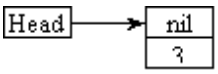
Определим запись типа TList с полями, содержащими характеристики данных – значения очередного элемента и адреса следующего за ним элемента

```
PList = ^TList;
  TList = record
     Data : Integer;
     Next : PList;
  end;
```

Чтобы список существовал, надо определить указатель на его начало. Опишем переменные.

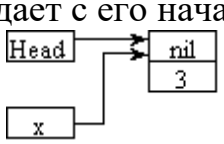
```
Var  Head, x : PList;
```

Создадим первый элемент: New(Head); { выделяем место в памяти для переменной Head } Head^.Next := nil; { указатель на следующий элемент пуст (такого элемента нет) } Head^.Data := 3; { заполняем информационное поле первого элемента }

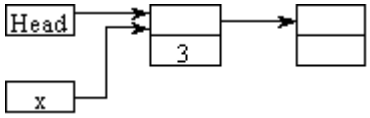


Продолжим формирование списка, для этого нужно добавить элемент в конец списка.

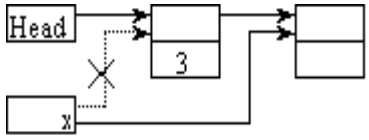
Введем вспомогательную переменную указательного типа, которая будет хранить адрес последнего элемента списка: x := Head; {сейчас последний элемент списка совпадает с его началом}



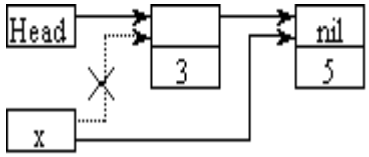
New(x^.Next); { выделим области памяти для следующего (2-го) элемента и поместим его адрес в адресную часть предыдущего (1-го) элемента }



x := x^.Next ; { переменная x принимает значение адреса выделенной области. Таким образом осуществляется переход к следующему (2-ому) элементу списка }



x^.Data := 5; { значение этого элемента } x^.Next := nil; {следующего значения нет }



Остальные числа заносятся аналогично: New(x^.Next); { выделим области памяти для следующего элемента } x := x^.Next ; { переход к следующему (3-му) элементу списка } x^.Data := 1; { значение этого элемента } x^.Next := nil; {следующего значения нет } New(x^.Next); { выделим области памяти для следующего элемента } x := x^.Next ; {

переход к следующему (4-му) элементу списка } x^.Data := 9; { значение этого элемента } x^.Next := nil; {следующего значения нет }

Замечание. Как видно из примера, отличным является только создание первого (Head) элемента – головы списка. Все остальные действия полностью аналогичны и их естественно выполнять в цикле.

Присоединение нового элемента к голове списка производится аналогично: …….. New(x); { ввод значения элемента x^.Data := … } x^.Next := Head; Head := x; ……..

В этом случае последний введенный элемент окажется в списке первым, а первый – последним.

Просмотр списка.

Просмотр элементов списка осуществляется последовательно, начиная с его начала. Указатель List последовательно ссылается на первый, второй и т. д. элементы списка до тех пор, пока весь список не будет пройден. При этом с каждым элементом списка выполняется некоторая операция– например, печать элемента. Начальное значение List – адрес первого элемента списка (Head). Digit – значение удаляемого элемента.

```
List := Head;
While List^.Next <> nil do
 begin
   WriteLn(List^.Data);
        List := List^.Next; { переход к следующему элементу; аналог
        для     массива i:=i+1 }
 end;
```

Пример вывода на экран списка в C++

```
void s_print(struct num **ptr)
{
   struct num *current=NULL;
   current=*ptr;
   if (current==NULL)
      printf("It`s empty\n");
        else{
          while(current != NULL){
             printf("%d ",current->digit);
             current  = current->next;
            }}
   printf("\n");
}
```

Удаление элемента из списка.

При удалении элемента из списка необходимо различать три случая:1. Удаление элемента из начала списка.

2. Удаление элемента из середины списка.

3. Удаление из конца списка.

Удаление элемента из начала списка.

```
List := Head; { запомним адрес первого элемента списка }
Head := Head^.List; { теперь Head указывает на второй элемент списка }
Dispose(List); { освободим память, занятую переменной List^ }
```

Удаление элемента из середины списка.

Для этого нужно знать адреса удаляемого элемента и элемента, находящегося в списке перед ним.

```
List := Head;
While (List<>nil) and (List^.Data<>Digit) do
 begin
   x := List;
   List := List^.Next;
 end;
x^.Next := List^.Next;
Dispose(List);
```

Удаление из конца списка.

Оно производится, когда указатель x показывает на предпоследний элемент списка, а List – на последний.

```
List := Head; x := Head;
While List^.Next<>nil do
 begin
   x := List;
   List := List^.Next;
 end;
x^.Next := nil;
Dispose(List);
```

Код C++ для удаления первого элемена из списка:

```
void delFirst(int info, spisok &sp)

{ elem *temp =sp.begin;
 if (sp.begin){
   sp.begin=sp.begin->next;
   if (sp.begin) sp.begin->prev=NULL;
   delete temp;
   sp.count--;
 }
```

}

Контрольные вопросы

1. Что такое указатели? Какие значения они могут принимать? Какие операции возможны над указателями?
2. Что представляют собой динамические структуры данных? Для чего они используются? Чем отличаются от данных статического типа?
3. Какие стандартные процедуры существуют в языке Pascal, C++ для работы с указателями?
4. Зачем различать типы указателей?
5. Какие операции требуется выполнить для вставки и удаления элемента списка?
6. Сколько элементов может содержать список?
7. Можно ли для построения списка обойтись одной переменной?

Варианты заданий

Вариант 1	Сформировать список строк и а) сохранить его в текстовом файле; б) сохранить его в обратном порядке в текстовом файле. Использовать рекурсию.
Вариант 2	Написать функцию, которая проверяет, упорядочены ли элементы списка по алфавиту
Вариант 3	Написать функцию, которая вычисляет среднее арифметическое элементов непустого списка.
Вариант 4	Написать процедуру присоединения списка L2 к списку L1.
Вариант 5	Написать функцию, которая создает список L2, являющийся копией списка L1, начинающегося с данного узла.
Вариант 6	Написать функцию, которая подсчитывает количество вхождений ключа в списке.
Вариант 7	Написать функцию, которая удаляет из списка все вхождения ключа.
Вариант 8	Сформировать список целых чисел и удалить из него все четные
Вариант 9	Написать функцию, подсчитывающую количество слов в списке, которые начинаются с той же буквы, что и следующее слово.
Вариант 10	Сформировать список целых чисел и упорядочить их по неубыванию.

Литература основная

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
3. Павловская, Т. А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.

Лабораторная работа №15
Стек

Цель работы:

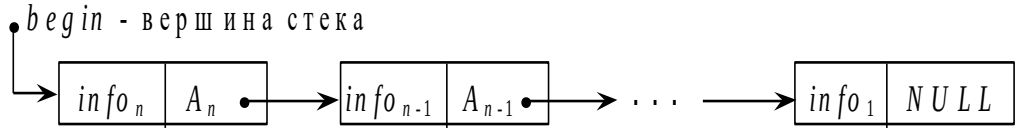
Закрепить технику работы с динамическими структурами данных на примере стека. Рассмотреть основные операции со стеком и познакомится с типичными примерами применения стека.

Оборудование, технические и инструментальные средства:

Персональный компьтер типа IBM PC,операционная система Windows XP/7/8,программная среда Borland Pascal,Microsoft Visual C++ ,стандартные библиотеки используемых языков программирования

Общие сведения. Структура данных СТЕК

Стек – упорядоченный набор элементов, в котором добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека, т.е. стек – это список с одной точкой доступа к его элементам. Графически это можно изобразить так:



В любой момент времени доступен лишь один элемент стека – верхний. Извлекать элементы из стека можно только в порядке, обратном их добавления в стек (первым пришел, последним ушел).

Значением указателя, представляющего стек, является ссылка на вершину стека, каждый элемент стека содержит поле ссылки на следующий элемент.

Стек – структура типа ***LIFO*** (*Last In, First Out*) – последним вошел, первым выйдет. Стек получил свое название из-за схожести с оружейным магазином с патронами (обойма): когда в стек добавляется новый элемент, то прежний проталкивается вниз и временно становится недоступным. Когда же верхний элемент удаляется из стека, следующий за ним поднимается вверх и становится опять доступным.

Максимальное число элементов стека ограничивается, т.е. по мере вталкивания в стек новых элементов память под него должна динамически запрашиваться и освобождаться также динамически при удалении элемента из стека. Таким образом, стек – динамическая структура данных, состоящая из переменного числа элементов одинакового типа.

Состояние стека рассматривается только по отношению к его вершине, а не по отношению к количеству его элементов, т.е. только вершина стека характеризует его состояние.

Операции, выполняемые над стеком, имеют специальные названия:

push – добавление элемента в стек (вталкивание);

pop – выталкивание (извлечение) элемента из стека, верхний элемент стека удаляется (не может применяться к пустому стеку).

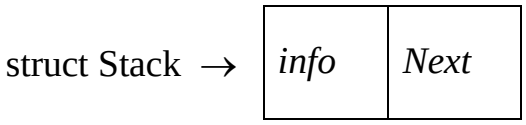
Кроме этих обязательных операций часто нужно прочитать значение элемента в вершине стека, не извлекая его оттуда. Такая операция получила название *peek*.

Примеры

Алгоритм формирования стека

Рассмотрим данный алгоритм для первых двух элементов.

1. Описание структуры переменной, содержащей информационное и адресное поля:



Шаблон структуры рекомендуется описывать глобально:

```
struct Stack {
        int info;
        Stack *Next;
} ;
```

2. Объявление указателей на структуру:

Stack *begin (вершина стека), *t (текущий элемент);

3. Так как первоначально стек пуст: begin = NULL;

4. Захват памяти под первый (текущий) элемент:

t = (Stack*) malloc (sizeof(Stack)); или t = new Stack;

формируется конкретный адрес ОП (обозначим его *A*1) для первого элемента, т.е. *t* равен *A*1.

5. Ввод информации (например, *i*1);

а) формирование информационной части:

t -> info = i1;

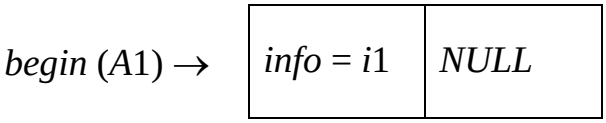
б) формирование адресной части: значение адреса вершины стека записываем в адресную часть текущего элемента (там был *NULL*)

t -> Next = begin;

6. Вершина стека переносится на созданный первый элемент:

begin = t;

в результате получается следующее:



7. Захват памяти под второй элемент:

t = (Stack*) malloc (sizeof(Stack)); или t = new Stack;

формируется конкретный адрес ОП (*A*2) для второго элемента.

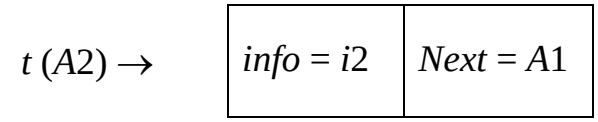
8. Ввод информации для второго элемента (*i*2);

а) формирование информационной части:

t -> info = i2;

б) в адресную часть записываем значение адреса вершины, т.е. адрес первого (предыдущего) элемента (*A*1):

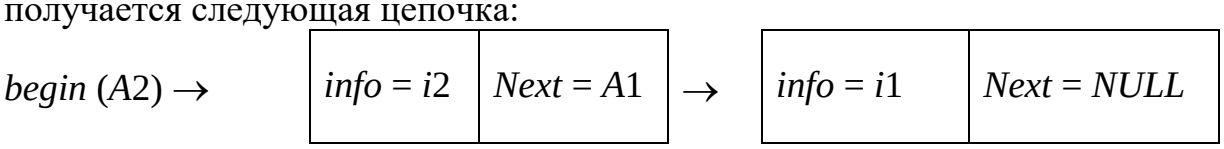
t -> Next = begin;



9. Вершина стека снимается с первого и устанавливается на новый элемент (*A*2):

begin = t;

получается следующая цепочка:



Обратите внимание, что действия 7, 8, 9 идентичны действиям 4, 5, 6, т.е. добавление новых элементов в стек можно выполнять в цикле, до тех пор, пока это необходимо.

Функция формирования элемента стека для объявленного ранее типа данных может выглядеть следующим образом:

```
Stack* Create(Stack *begin) {
    Stack *t = (Stack*)malloc(sizeof(Stack));
    printf("\n Input Info  ");
    scanf("%d", &t -> info);
    t -> Next = begin;
    return t;
    }
```

Участок программы с обращением к функции *Create* для добавление необходимого количества элементов в стек может иметь следующий вид на C++:

```
Stack *begin = NULL;
int repeat = 1;
while(repeat)  {              // repeat=1 – продолжение ввода данных
    begin = Create(begin);
    printf(" Stop - 0 ");     // repeat=0 – конец ввода данных
    scanf("%d", &repeat);
}
```

Занесение в стек	*Извлечение элемента из стека.*
Var x : PStack; …… New(x); x^.Data := …….; x^.Next := Stack; Stack := x; ……	Var N : Integer; x : PStack; …… N := Stack^.Data; x:= Stack; Stack := Stack^.Next; Dispose(x);

Занесение в стек производится аналогично вставке нового элемента в начало списка.

Извлечение элемента из стека.

В результате выполнения этой операции некоторой переменной N должно быть присвоено значение первого элемента стека и изменено значение указателя на начало стека: Контрольные вопросы

1. В чем сходство и отличие динамических структур данных типа список и стек?
2. Можно ли добраться до середины или конца («дна») стека, минуя его начало («вершину»)?
3. Приведите примеры из жизни, где встречается «принцип стека».
4. Оформите в виде процедуры Push занесение в стек значения.
5. Оформите в виде процедуры Pop извлечение значения из стека.

Варианты заданий для составления программ

Вариант 1	Подсчитать количество элементов в стеке.
Вариант 2	Сформировать стек, содержащий строки и сохранить его в текстовом файле.
Вариант 3	Восстановить стек, содержащий строки, из текстового файла.
Вариант 4	Написать функцию, которая вычисляет среднее арифметическое элементов стека.
Вариант 5	Написать процедуру присоединения стека S2 к стеку S1.
Вариант 6	Определить симметричность произвольного текста любой длины. Текст должен оканчиваться точкой. Задачу решить с помощью двух стеков.
Вариант 7	Слить два стека, содержащих возрастающую последовательность целых положительных чисел, в третий стек так, чтобы его элементы располагались также в порядке возрастания.
Вариант 8	В данном тексте проверить соответствие открытия и закрытия скобок.
Вариант 9	Напечатать содержимое текстового файла, выписывая символы каждой его строки в обратном порядке.
Вариант 10	Проверить, является ли строка палиндромом.

ПРИЛОЖЕНИЕ 1. Таблицы символов ASCII

Стандартная часть таблицы символов ASCII

КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С
0		16	►	32		48	0	64	@	80	P	96	`	112	p
1	☺	17	◄	33	!	49	1	65	A	81	Q	97	A	113	q
2	☻	18	↕	34	"	50	2	66	B	82	R	98	B	114	r
3	♥	19	‼	35	#	51	3	67	C	83	S	99	C	115	s
4	♦	20	¶	36	$	52	4	68	D	84	T	100	D	116	t
5	♣	21	§	37	%	53	5	69	E	85	U	101	E	117	u
6	♠	22	▬	38	&	54	6	70	F	86	V	102	F	118	v
7	•	23	↨	39	'	55	7	71	G	87	W	103	G	119	w
8	◘	24	↑	40	(	56	8	72	H	88	X	104	H	120	x
9	○	25	↓	41	)	57	9	73	I	89	Y	105	I	121	Y
10	◙	26	→	42	*	58	:	74	J	90	Z	106	J	122	Z
11	♂	27	←	43	+	59	;	75	K	91	[	107	K	123	{
12	♀	28	∟	44	,	60	<	76	L	92	\	108	L	124	|
13	♪	29	↔	45	-	61	=	77	M	93	]	109	M	125	}
14	♫	30	▲	46	.	62	>	78	N	94	^	110	N	126	~
15	☼	31	▼	47	/	63	?	79	O	95	_	111	O	127	⌂

Некоторые из вышеперечисленных символов имеют особый смысл. Так, например, символ с кодом 9 обозначает символ горизонтальной табуляции, символ с кодом 10 – символ перевода строки, символ с кодом 13 – символ возврата каретки.

Дополнительная часть таблицы символов

КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С	КС	С
128	А	144	Р	160	а	176	▒	192	└	208	╨	224	Р	240	Ё
129	Б	145	С	161	б	177	▒	193	┴	209	╤	225	С	241	ё
130	В	146	Т	162	в	178	▓	194	┬	210	╥	226	Т	242	Є
131	Г	147	У	163	г	179	│	195	├	211	╙	227	У	243	є
132	Д	148	Ф	164	д	180	┤	196	─	212	╘	228	Ф	244	Ї
133	Е	149	Х	165	е	181	╡	197	┼	213	╒	229	Х	245	ї
134	Ж	150	Ц	166	ж	182	╢	198	╞	214	╓	230	Ц	246	Ў
135	З	151	Ч	167	з	183	╖	199	╟	215	╫	231	Ч	247	ў
136	И	152	Ш	168	и	184	╕	200	╚	216	╪	232	Ш	248	°
137	Й	153	Щ	169	й	185	╣	201	╔	217	┘	233	Щ	249	·
138	К	154	Ъ	170	к	186	║	202	╩	218	┌	234	Ъ	250	·
139	Л	155	Ы	171	л	187	╗	203	╦	219	█	235	Ы	251	√
140	М	156	Ь	172	м	188	╝	204	╠	220	▄	236	Ь	252	№
141	Н	157	Э	173	н	189	╜	205	═	221	▌	237	Э	253	¤
142	О	158	Ю	174	о	190	╛	206	╬	222	▐	238	Ю	254	■
143	П	159	Я	175	п	191	┐	207	╧	223	▀	239	Я	255	

В таблицах обозначение **КС** означает «код символа», а **С** – «символ».

ПРИЛОЖЕНИЕ 2. Операции ЯП С/С++

Операции приведены в порядке убывания приоритета, операции с разными приоритетами разделены чертой.

Опера-ция	Краткое описание	Использование	Порядок выполнения
Первичные (унарные) операции			
.	Доступ к члену	*объект* **.** *член*	Слева Направо
->	Доступ по указателю	*указатель* -> *член*	
[ ]	Индексирование	*переменная* [*выражение*]	
( )	Вызов функции	ID_*функции*(*список*)	
Унарные операции			
++	Постфиксный инкремент	*lvalue*++	Справа Налево
--	Постфиксный декремент	*lvalue*--	
sizeof	Размер объекта (типа)	sizeof(ID *или тип*)	
++	Префиксный инкремент	++*lvalue*	
--	Префиксный декремент	--*lvalue*	
~	Побитовое НЕ	~*выражение*	
!	Логическое НЕ	!*выражение*	
– (+)	Унарный минус (плюс)	– (+)*выражение*	
*****	Разадресация	**выражение*	
&	Адрес	&*выражение*	
()	Приведение типа	(*тип*)*выражение*	
Бинарные и тернарная операции			
*****	Умножение	*выражение* * *выражение*	Слева Направо
/	Деление	*выражение* / *выражение*	
%	Получение остатка	*выражение* % *выражение*	
+	Сложение	*выражение* + *выражение*	

–	Вычитание	*выражение* – *выражение*	
<<	Сдвиг влево	*выражение* << *выражение*	
>>	Сдвиг вправо	*выражение* >> *выражение*	
<	Меньше	*выражение* < *выражение*	
<=	Меньше или равно	*выражение* <= *выражение*	
>	Больше	*выражение* > *выражение*	
>=	Больше или равно	*выражение* >= *выражение*	
==	Равно	*выражение* == *выражение*	
!=	Не равно	*выражение* != *выражение*	
&	Побитовое И	*выражение* & *выражение*	
^	Побитовое исключ. ИЛИ	*выражение* ^ *выражение*	

Опера-ция	Краткое описание	Использование	Порядок выполнения
|	Побитовое ИЛИ	*выражение* | *выражение*	Слева направо
&&	Логическое И	*выражение* && *выражение*	
||	Логическое ИЛИ	*выражение* || *выражение*	
?:	Условная операция **(*тернарная*)**	*выражение* ? *выражение* : *выражение*	
=	Присваивание	*lvalue* = *выражение*	
***=**	Умножение с присваиванием	*lvalue* *= *выражение*	

/=	Деление с присваиванием	*lvalue* /= *выражение*	
%=	Остаток от деления с присваиванием	*lvalue* %= *выражение*	
+=	Сложение с присваиванием	*lvalue* += *выражение*	
- =	Вычитание с присваиванием	*lvalue* -= *выражение*	
<<=	Сдвиг влево с присваиванием	*lvalue* <<= *выражение*	Справа Налево
>>=	Сдвиг вправо с присваиванием	*lvalue* >>= *выражение*	
&=	Поразрядное И с присваиванием	*lvalue* &= *выражение*	
|=	Поразрядное ИЛИ с присваиванием	*lvalue* |= *выражение*	
^=	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ с присваиванием	*lvalue* ^= *выражение*	
,	Последовательное вычисление	*выражение*, *выражение*	Слева направо

ПРИЛОЖЕНИЕ 3. Стандартные математические функции

Математические функции языка Си декларированы в файлах ***math.h*** и ***stdlib.h.*** В приведенных здесь функциях аргументы и возвращаемый результат имеют тип *double*. Аргументы тригонометрических функций должны быть заданы в радианах (2π радиан = 360°).

Название	Обозначение	*ID* функции в языке Си
квадратный корень	$\sqrt{x}$	sqrt(*x*)
абсолютная величина	$\|x\|$	fabs(*x*)
экспонента	e^x	exp(*x*)
степенная функция	x^y	pow(*x*,*y*)
натуральный логарифм	ln(*x*)	log(*x*)
десятичный логарифм	$lg_{10}(x)$	log10(*x*)
синус аргумента x	sin(*x*)	sin(*x*)
косинус аргумента x	cos(*x*)	cos(*x*)
тангенс	tg(*x*)	tan(*x*)
арксинус	arcsin(*x*)	asin(*x*)
арккосинус	arccos(*x*)	acos(*x*)
арктангенс	arctg(*x*)	atan(*x*)
гиперболический синус	$sh(x)=0.5\ (e^x-e^{-x})$	sinh(*x*)
гипербол.косинус	$ch(x)=0.5\ (e^x+e^{-x})$	cosh(*x*)
гиперболическ.тангенс	tgh(*x*)	tanh(*x*)
остаток от деления *x* на *y*		fmod(*x*,*y*)
наименьшее целое >=*x*		ceil(*x*)
наибольшее целое <=*x*		floor(*x*)

Список рекомендуемой литературы

Основная литература

1. Иванова Г.С. Технология программирования Учебник для вузов – М,МГТУ им.Баумана,2002
2. Климова Л.М. Pascal 7.0.:Практическое программирование. Решение типовых задач.-2 изд.,доп.-М.:Кудиц-Образ,2000.528 с.
3. Немнюгин С. А. Turbo Pascal;практикум./МОРФ.-СПб:Питер,2003.-256с.:ил.
4. Немнюгин С.А. Turbo PASCAL:Практикум:Учебное пособие/ МОРФ.-2 изд.-М.:СПб.,2005.-267 с.
5. Малыхина М. П. Pascal 7.0:Практическое программирование.Решение типовых задач:Учебное пособие.-М.:Кудиц-Образ,2000-528 с.
6. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – Минск : Бестпринт, 2001
7. Павловская, Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
8. Павловская, Т. А C/C++. Структурное программирование : Практикум /. C/C++. Структурное программирование : Практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2002.
9. Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.
10. Климова, Л. И. C++. Практическое программирование / Л. И. Климова. – М. : Кудиц-Образ, 2001.

Дополнительная литература

1. Фаронов. В.В. Программирование на языке высокого уровня Turbo Pascal:Учебное пособие/МОРФ.-СПб.:БХВ-Петербург,2006. 523 с.
2. Меженный О. А. Turbo Pascal:Учебное пособие/МОРФ.-СПб.:Питер,2007.-366 с.
3. Алексеев Е. Р. Турбо Паскаль 7.0.:Начальный курс.Учебное пособие.-7-е изд.,перераб.-М.:Нолидж,2000.-576с.:ил.
4. Е.Р.Алексеев Турбо Паскаль7.0. Е.Р.Алексеев,Чеснокова О.В.,Павлыш В.Н.,Славинская Л.В.-Москва:Изд-во АСТ,2004.-270с :ил.
5. Архангельский, А. Я. Программирование в C++ Builder 6 / А. Я. Архангельский. – М. : ЗАО «Издательство БИНОМ», 2002.
6. Березин, Б. И. Начальный курс C и C++ / Б. И. Березин, С. Б. Березин. – М. : Диалог– МРТИ, 1999.
7. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. – СПб. : Невский диалект, 2001.
8. Вирт, Н. Алгоритмы + структуры данных = программы / Н. Вирт. – М. : Мир, 1985.

9. Касаткин, А. И. Профессиональное программирование на языке Си: От Turbo-C к Borland C++: справ.пособие / А. И. Касаткин, А. Н. Вольвачев. – Минск : Выш. шк., 1992.
10. Касаткин, А. И. Профессиональное программирование на языке Си. Управление ресурсами: справ.пособие / А. И. Касаткин. – Минск : Выш.шк., 1992.
11. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992.
12. Керниган, Б. Язык программирования Си. Задачи по языку Си / Б. Керниган, Д. Ритчи, А. Фьюэр. – М. : Финансы и статистика, 1985.
13. Керниган, Б. Универсальная среда программирования UNIX / Б. Керниган, Р. Пайк. – М. : Финансы и статистика, 1992.
14. Кнут, Д. Искусство программирования: т. 1–3. Основные алгоритмы / Д. Кнут. – М. : Издательский дом «Вильямс», 2004.
15. Котлинская, Г. П. Программирование на языке Си / Г. П. Котлинская, О. И. Галиновский. – Минск : Выш.шк., 1991.
16. Котов, В. М. Структуры данных и алгоритмы: учеб. пособие / В. М. Котов, Е. П. Соболевская. – Минск : БГУ, 1996.
17. Морозов, А. А. Структуры данных и алгоритмы: учеб. пособие : в 2 ч. / А. А. Морозов. – Минск : БГПУ им. М. Танка.
18. Подбельский, В. В. Программирование на языке Си / В. В. Подбельский, С. С. Фомин. – М. : Финансы и статистика, 2001.
19. Подбельский, В. В. Язык C++ / В. В. Подбельский. – М. : ФиС, 2001.
20. Пол, И. Объектно-ориентированное программирование с использованием C++ / И. Пол. – Киев : ДиаСофт, 1995.
21. Практикум по программированию на алгоритмических языках / Г. И. Светозарова [и др.]. – М. : Наука, 1980.
22. Культин Н. Самоучитель:Программирование в Turbo Pascal7.0 и Delphi.-2 изд.- Спб.:БХВ-Петербург,2003.-404 с