Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра инженерной кибернетики

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Методические указания по выполнению лабораторных работ для студентов специальности 5В070200

Алматы 2014

СОСТАВИТЕЛИ: Н.В. Сябина, Л.Н.Рудакова. Технологии программирования. Методические указания по выполнению лабораторных работ для студентов специальности 5В070200 - Автоматизация и управление. – Алматы: АУЭС, 2014. – 26 с.

Настоящие методические указания включают 8 лабораторных работ по дисциплине «Технологии программирования», которые позволят студентам получить практические навыки разработки программ на языке высокого уровня С++. Контрольные вопросы, приведенные в конце каждой лабораторной работы, помогут закрепить полученные теоретические знания.

В приложении содержится необходимый справочный материал.

Методические указания предназначены для студентов всех форм обучения специальности 5В070200 - Автоматизация и управление.

Табл. 9, библиогр. – 6 назв.

Рецензент: канд. техн. наук, старший преподаватель Г.Д. Мусапирова

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2014 г.

1 Лабораторная работа. Использование базовых и дополнительных структур С++ при решении вычислительных задач

Цель: научиться практически применять знания, полученные в курсе Информатики, к решению задач вычислительного характера на языке высокого уровня С++.

1.1 Общие сведения

1.1.1 Любой сложный алгоритм можно представить, используя три основные управляющие конструкции. К *базовым* относят:

а) *следование* - обозначает последовательное выполнение действий;

б) *ветвление* - выбор одного из двух вариантов действий;

в) *цикл-пока* - определяет повторение действий, пока не будет нарушено некоторое условие, выполнение которого проверяется в начале цикла.

Кроме базовых, процедурные языки программирования высокого уровня используют три *дополнительные* конструкции, реализуемые через базовые:

а) *выбор* - выбор одного варианта из нескольких в зависимости от значения некоторой величины;

б) *цикл-до* - повторение действий до выполнения заданного условия, проверка которого осуществляется после выполнения действий в цикле;

в) *цикл с заданным числом повторений* (счетный цикл) - повторение некоторых действий указанное количество раз.

Перечисленные конструкции были положены в основу структурного программирования, а в языках программирования высокого уровня появились управляющие операторы для их реализации.

Для изображения алгоритмов структурных программ могут использоваться:

1) *блок-схемы*;

2) *псевдокоды*;

3) *Flow-формы*;

4) *диаграммы Насси-Шнейдермана*.

1.1.2 Программы, написанные с использованием только структурных операторов передачи управления, называют *структурными,* чтобы подчеркнуть их отличие от программ, разрабатываемых с использованием низкоуровневых способов передачи управления.

Каждой управляющей конструкций, рассмотренной в п.1.1.1, можно поставить в соответствие оператор С++. Так, например, к базовым следует отнести:

а) *оператор присваивания* переменная = выражение;

б) оператор *условного перехода*:

- в полной форме if (условие) оператор1; else оператор2;
- в краткой форме if (условие) оператор;

в) *оператор цикла с предварительным условием* (с *предусловием*)

```
while (условие)          или          while (условие)
{                                         оператор;
оператор1;
оператор2;
операторN;
}
```

Дополнительным конструкциям следует поставить в соответствие:

а) оператор-*переключатель switch* (оператором *выбора* или *варианта*)

```
switch (выражение)
{
case n1 : оператор1; break;
case n2 : оператор2; break;
case nk : операторК; break;
default : операторN; break;
}
```

б) *троичный условный* оператор

```
(выражение1) ? выражение2: выражение3;
```

в) *оператор цикла с последующим условием* (с *постусловием*)

```
do                       или        do {
{оператор;}                         оператор1;
while (условие);                    оператор2;
                                    …
                                    операторN;
                                    } while (условие);
```

г) *оператор цикла с параметром*

```
for (инициализация; проверка; приращение)
{оператор1;
оператор2;
…
операторN; }
```

1.2 Задание к лабораторной работе

Выбрать вариант по таблице 1.1 и решить задачу, используя базовые и дополнительные структуры. К каждой задаче построить блок-схему. Организовать диалог с пользователем.

Таблица 1.1 – Варианты заданий

Вариант	Задание
Вычислить:	
1	Сопротивление проводника, если известны (задаются пользователем): удельное сопротивление ρ, площадь поперечного сечения S, а длина проводника L изменяется от 1 до 20 см.

2	Все смещения точки при распространении незатухающих колебаний заданной амплитудой A, периодом колебаний T и скоростью распространения c, отстоящей от источника колебаний на расстоянии L при изменении t от 0 до 10 сек с шагом 0,5 сек.
3	Высоту поднятия жидкости в капиллярной трубке, если заданы радиус трубки r, плотность жидкости ρ, коэффициент поверхностного натяжения α, а краевой угол θ изменяется от 0 до π.
4	Наибольший общий делитель двух целых чисел.
5	Все смещения точки, совершающей гармонические колебания с заданной амплитудой A, периодом колебаний T и начальной фазой φ, от положения равновесия при изменении t от 0 до 10 сек с шагом 0,5 сек.
Вывести на экран:	
6	Работающие «электронные часы», которые функционируют до нажатия любой клавиши.
7	Таблицу умножения на число n, задаваемое пользователем.
8	Таблицу квадратов m первых целых положительных чисел.
9	Таймер, который по истечении заданного промежутка времени t, величина которого вводится с клавиатуры, выдает звуковой сигнал.
10	Изображение шахматной доски: черные клетки отображать «звездочкой», белые - пробелом.
11	Таблицу степеней (от нулевой до k-ой) числа Z; количество степеней k задается пользователем.
Проверить:	
12	Является ли функция $y = \frac{1}{x}\sin x$ периодической, если аргумент x изменяется в интервале [0;5T], а период T=2π?
13	Знание пользователем таблицы умножения: вывести k примеров и выставить оценку (90-100% правильных ответов - «отлично», 75-89% - «хорошо», за 55-74% - «удовлетворительно», менее 55% - «плохо»).
14	Предложить пользователю угадать сгенерированное компьютером целое число в диапазоне от 1 до 10 за 5 попыток.
15	Являются ли k целых чисел, введенных пользователем, простыми?

1.3 Контрольные вопросы

1.3.1 Какие алгоритмические структуры относятся к базовым? Какие к дополнительным? Приведите примеры базовых и дополнительных структур.

1.3.2 Как графически изображаются схемы алгоритмов? Какое преимущество имеют Flow-формы и диаграммы Насси-Шнейдермана? Какие достоинства и недостатки имеют блок-схемы?

1.3.3 В чем отличие между оператором и операцией? Какие виды операций существуют?

1.3.4 Перечислите известные Вам операции присваивания в С++.

1.3.5 Какие операторы С++ реализуют ветвление? В чем их особенности? Приведите примеры использования.

1.3.6 Какую структуру имеет оператор выбора? С какой целью используется оператор break?

1.3.7 В чем заключаются особенности форматного ввода – вывода?

1.3.8 Какие операторы С++ реализуют циклы? В чем их особенности? Приведите примеры использования.

1.3.9 Приведите примеры организации пустого и бесконечного циклов.

1.3.10 Как с помощью цикла while можно сымитировать цикл for?

2 Лабораторная работа. Характерные приемы программирования

Цель: получить практические навыки работы со стандартными алгоритмами – характерными приемами программирования в С++, а также навыки использования численных методов.

2.1 Общие сведения

2.1.1 Часто при решении разного рода вычислительных задач приходится использовать одни и те же стандартные алгоритмы для получения суммы, произведения, количества элементов некоторой структуры, поиска максимального или минимального их значения, так называемые характерные приемы программирования. На практике реализация всех характерных приемов сводится к следующему:

- до открытия цикла задается начальное значение накапливаемого или предположительное значение искомого параметра;
- внутри цикла осуществляется непосредственно накапливание или поиск.

Рекомендации по практическому осуществлению характерных приемов программирования приведены в таблице 2.1.

Таблица 2.1 – Характерные приемы программирования

Прием программирования	Действия, выполняемые до цикла	Действия, выполняемые в цикле
Накапливание *суммы*	S = 0;	S+=*элемент_массива*;
Накапливание *произведения*	P = 1;	P*=*элемент_массива*;
Накапливание *количества*	k = 0;	k++;

Поиск *максимального* значения	max=*предполаг_знач*;	if (*текущ_элемент*>max) max=*текущ_элемент*;
Поиск *минимального* значения	min=*предполаг_знач*;	if (*текущ_элемент* <min) min=*текущ_элемент*;

2.1.2 Часто в математических вычислениях возникает необходимость определить *сумму бесконечного ряда с заданной точностью*. На такие задачи налагается жесткое условие: ряд должен быть сходящимся, т.е. абсолютное значение текущего элемента ряда должно быть меньше абсолютного значения его предыдущего элемента.

В подобных случаях можно пользоваться итерационными циклами, реализуемыми операторами цикла с предварительным и с последующим условиями. Иначе условие выхода из цикла никогда не будет выполнено и цикл станет бесконечным.

2.1.3 Если определен способ вычисления текущего значения заданной последовательности по предыдущему значению, то говорят, что задана *рекуррентная формула* вычисления элемента последовательности. В таком способе задания есть необходимый атрибут – явно заданное начальное значение этой последовательности.

При вычислении элементов рекуррентно заданной последовательности может использоваться любой из операторов цикла в зависимости от поставленной задачи.

2.2 Задание к лабораторной работе

Выбрать вариант по таблице 2.1 и решить задачу, используя известные численные методы и характерные приемы программирования. К каждой задаче построить блок-схему. Организовать диалог с пользователем.

Таблица 2.2 – Варианты заданий

Вариант	Задание
Вычислить:	
1	Среднее арифметическое последовательности дробных чисел, вводимой с клавиатуры; количество чисел должно задаваться пользователем.
2	Момент времени t, в который будет достигнута максимальная скорость точки, если уравнение движения точки дано в виде: $x = 2\sin(\frac{\pi}{2}t + \frac{\pi}{4})$.
3	Число π с заданной пользователем точностью, для чего воспользоваться числовым рядом $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \ldots$, значение частичной суммы которого при суммировании достаточно большого количества членов приближается к значению π/4.

4	Минимальное значение среди сгенерированной последовательности из k случайных чисел в диапазоне от 0 до 100, вывести эти числа на экран.
5	Факториал числа, введенного с клавиатуры.
6	Среднее геометрическое среди сгенерированной последовательности из 10 случайных чисел в диапазоне от 1 до 10, вывести эти числа на экран.
7	Через сколько лет арендатором накопится сумма S, достаточная для покупки собственного помещения, если его стартовый капитал – k тенге, ежемесячный доход – n%, аренда помещения – m тенге.
8	Максимальное значение среди сгенерированной последовательности из k случайных чисел в диапазоне от 0 до 50, вывести эти числа на экран.
9	Момент времени t, в который будет достигнуто максимальное ускорение точки, если уравнение движения точки дано в виде: $x = \sin(\frac{\pi}{4}t + \frac{\pi}{8})$.

Численно убедиться в справедливости равенства, для чего для заданного пользователем значения аргумента x вычислить левую его часть и разложение, стоящее в правой части с заданной погрешностью:

10	$\ln(1-x) = -(x + \frac{x^2}{2} + \frac{x^3}{3} + ... + \frac{x^n}{n} + ...,\ x < 1.$
11	$\sin x = x\left(1 - \frac{x^2}{\pi^2}\right)\left(1 - \frac{x^2}{4\pi^2}\right)...\left(1 - \frac{x^2}{(n-1)^2\pi^2}\right)...$
12	$arctgx = x - \frac{x^3}{3} + \frac{x^5}{5} - ... + (-1)^n \frac{x^{2n+1}}{2n+1} + ...$
13	$\frac{\pi^2}{8} - \frac{\pi}{4}\lvert x\rvert = \frac{\cos 3x}{3^2} + \frac{\cos 5x}{5^2} + ... + \frac{\cos(2n+1)x}{(2n+1)^2} + ...,\ \lvert x\rvert < 1.$
14	$x = 2\left(\sin x - \frac{\sin 2x}{2} + \frac{\sin 3x}{3} - ... + (-1)^{n-1}\frac{\sin nx}{n} + ...\right),\ -\pi < x < \pi.$
15	$a^x = 1 + \frac{x\ln a}{1!} + \frac{(x\ln a)^2}{2!} + ... + \frac{(x\ln a)^n}{n!} + ...$

2.3 Контрольные вопросы

2.3.1 Какие вычислительные процессы считаются итерационными?
2.3.2 Какие формулы называются рекуррентными?
2.3.3 Что такое аппроксимация функций?
2.3.4 Каковы преимущества численных методов интегрирования?

2.3.5 В каких случаях целесообразно применение численных методов интегрирования?

2.3.6 Почему при накоплении количества, суммы или произведения начальное значение целесообразно инициализировать до цикла?

2.3.7 Приведите алгоритм поиска максимального значения среди элементов последовательности $a_1, a_2, a_3 \ldots a_{10}$.

2.3.8 Как выбирается предполагаемое значение при поиске минимального значения?

2.3.9 Приведите алгоритм поиска среднего арифметического элементов последовательности $a_1, a_2, a_3 \ldots a_{10}$.

2.3.10 Чем будет отличаться поиск среднего геометрического элементов приведенной в п.2.3.8 последовательности?

3 Лабораторная работа. Массивы и их обработка

Цель: получить практические навыки работы с массивами и представления о способах их обработки.

3.1 Общие сведения

Массив — это упорядоченный набор элементов одинакового типа, имеющих общее имя. Все элементы массива пронумерованы. Порядковый номер элемента в массиве называется *индексом.* Массивы должны быть обязательно описаны перед использованием в программе. Различают одномерные (*векторы)*, двумерные *(матрицы)* и многомерные массивы, однако на практике массивы размерностью свыше трех используются крайне редко. Общий вид объявления массива:

тип_элементов имя_массива [N1][N2]...[Nk];

Количество индексов [N1][N2]...[Nk] определяет размерность массива. При объявлении массива указывается общее число элементов массива. Индексация элементов массива в С++ по умолчанию начинается с нуля. Размер массива может задаваться константой или константным выражением. Нельзя задать массив переменного размера, для этой цели используется отдельный механизм - *динамическое выделение памяти*. В языке С++ возможна инициализация массива при его объявлении.

К элементам массива применимы две операции: *присваивание* и *выборка*. Для выборки элементов массива и их обработки чаще всего используются *циклы*.

3.2 Задание к лабораторной работе

Решить задачу, соответствующую варианту, выбранному по таблице 3.1. Построить блок-схему программы и организовать ввод данных, если это необходимо, с помощью генератора случайных чисел.

Таблица 3.1 – Варианты заданий

Вариант	Задание
1	Дана целочисленная прямоугольная матрица. Определить: количество строк, не содержащих ни одного нулевого элемента; максимальное из чисел, встречающихся в заданной матрице более одного раза.
2	Дана целочисленная прямоугольная матрица, содержащая элементы, равные 1. Определить количество столбцов, не содержащих ни одного элемента равного 1; среднее геометрическое элементов строки, содержащей минимальный элемент побочной диагонали.
3	Дана целочисленная прямоугольная матрица. Определить: количество столбцов, содержащих хотя бы один нулевой элемент; номер строки, в которой находится самая длинная серия одинаковых элементов.
4	Дана целочисленная прямоугольная матрица. Определить: произведение элементов в тех строках, которые не содержат отрицательных элементов; максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.
5	Дана целочисленная прямоугольная матрица. Определить: сумму элементов в тех столбцах, которые не содержат положительных элементов; минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.
6	Дана целочисленная прямоугольная матрица. Матрица А имеет седловую точку $A_{ij,}$, если A_{ij} является минимальным элементом в i–той строке и максимальным в j-столбце. Определить: сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.
7	Для заданной матрицы размером 10x10 найти такие k, что k-я строка совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.
8	Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.
9	Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Подсчитать количество локальных минимумов заданной матрицы размером 10x10.
10	Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.

11	Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.
12	Переставить элементы квадратной целой матрицы так, чтобы ее минимальный элемент находился в левом верхнем углу, следующий по величине - (1,1), следующий по величине – в позиции (2,2) и т.д., заполнив всю главную диагональ.
13	Переставить элементы квадратной вещественной матрицы так, чтобы ее максимальный элемент находился в левом верхнем углу, следующий по величине - (1,1), следующий по величине – в позиции (2,2) и т.д., заполнив всю главную диагональ.
14	Дана целочисленная квадратная матрица. Определить: среднее арифметическое элементов в тех строках, которые не содержат отрицательных элементов. Результат переписать в вектор.
15	Дан двумерный массив размерностью 4x7, заполненный целыми числами. Сформировать одномерный массив, каждый элемент которого равен наибольшему по модулю элементу соответствующего столбца.
16	Найти среднее геометрическое элементов, расположенных выше побочной диагонали вещественной матрицы 10x10. Переписать элементы, расположенные после минимального элемента в вектор.
17	Дан двумерный массив размерностью 6x5, заполненный целыми числами. Сформировать одномерный массив, каждый элемент которого равен количеству отрицательных элементов, кратных 2 или 4, соответствующей строки.
18	Элемент матрицы называется локальным максимумом, если он строго больше всех имеющихся у него соседей. Подсчитать количество локальных максимумов заданной матрицы размером 12x12.
19	Дан двумерный массив размерностью 6x5, заполненный целыми числами. Сформировать одномерный массив, каждый элемент которого равен первому четному элементу соответствующего столбца. Если такого нет, то он равен нулю.
20	Характеристикой строки целочисленной матрицы A 10x8 назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.
21	Дана целочисленная прямоугольная матрица. Матрица A имеет седловую точку $A_{ij,}$, если A_{ij} является минимальным элементом в i–той строке и максимальным в j-столбце. Определить: произведение элементов в тех строках, которые содержат хотя бы один положительный элемент.

22	Дан двумерный массив размерностью 9x5, заполненный целыми числами. Сформировать одномерный массив, каждый элемент которого равен наименьшему по модулю элементу соответствующего столбца.
23	Найти сумму элементов целочисленной матрицы 10x8 в тех столбцах, которые содержат хотя бы один отрицательный элемент. Переписать их в одномерный массив.
24	Найти среднее геометрическое модулей элементов целочисленной матрицы, расположенных на побочной диагонали. Определить максимальный элемент в матрице и его местоположение.
25	Дана целочисленная квадратная матрица. Определить минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы. Суммы переписать в одномерный массив.
26	Найти номер первой из строк вещественной матрицы, не содержащих ни одного отрицательного элемента. Определить максимальную сумму в столбцах матрицы.
27	Найти номер первого из столбцов целочисленной матрицы, не содержащих ни одного положительного элемента. Определить минимальное произведение в строках матрицы.
28	Найти сумму модулей элементов, расположенных выше главной диагонали целочисленной матрицы 10x10. Переписать элементы, расположенные после максимального элемента в вектор.
29	Найти произведение элементов в тех столбцах целочисленной матрицы, которые содержат хотя бы один положительный элемент. Переписать элементы, расположенные до минимального элемента в одномерный массив.
30	Дана целочисленная квадратная матрица. Определить: среднее геометрическое элементов в тех строках, которые не содержат отрицательных элементов. Максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

3.3 Контрольные вопросы

3.3.1 Что представляет собой массив элементов?

3.3.2 Какой максимальной размерности может быть многомерный массив?

3.3.3 Как объявляется массив?

3.3.4 С какой целью используется механизм выделения динамической памяти?

3.3.5 Как инициализируются элементы массива?

3.3.6 Как можно сформировать случайным образом массив элементов?

3.3.7 Какие способами ввода элементов массива существуют в C++?

3.3.8 Приведите алгоритм нахождения максимального значения в каждой строке матрицы произвольного размера.

3.3.9 Каковы особенности обработки квадратной матрицы?

3.3.10 Как организовать обработку матрицы произвольной размерности по столбцам (по строкам)?

4 Лабораторная работа. Методы сортировки массивов

Цель: получить практические навыки использования различных алгоритмов сортировки массивов в C++.

4.1 Общие сведения

Во многих задачах бывает необходимо переставить элементы массива таким образом, чтобы они располагались в порядке возрастания или убывания. Такие массивы называются *упорядоченными,* а процесс их получения — *сортировкой*.

4.1.1 Сортировка простым выбором.

Самым простым алгоритмом является *сортировка простым выбором*. В сортируемом массиве находится минимальный элемент, который обменивается местами с элементом в начале массива. Оставшаяся часть массива рассматривается как самостоятельный массив, в котором также производится поиск наименьшего элемента и его обмен с первым элементом укороченного массива. Действия повторяются до тех пор, пока массив не укоротится до одного элемента.

4.1.2 Метод пузырьковой сортировки.

Под *пузырьковой* сортировкой понимают целый класс алгоритмов сортировки. В своем простейшем варианте пузырьковая сортировка выполняется крайне медленно, поэтому обычно применяют пузырьковую сортировку с элементами оптимизации. Все алгоритмы пузырьковой сортировки имеют общую особенность – обмен элементов производится между двумя соседними элементами массива.

Можно уменьшить количество проходов сортировки, выполняя их не $(N-1)^2$ раз, а пока массив не будет отсортирован. Определить этот факт достаточно просто: если массив уже отсортирован, то в процессе прохода в нем не происходит никаких перестановок. Перед началом просмотра нужно установить признак отсутствия перестановок (*флаг*). В случае, если производится хотя бы одна перестановка, флаг изменяет свое значение. Если к моменту завершения прохода значение флага осталось первоначальным, значит, массив отсортирован и дальнейшие проходы не нужны.

Для оптимизации метода пузырьковой сортировки по времени выполнения каждого прохода можно использовать то, что после первого прохода наибольший элемент окажется в конце массива на предназначенном ему месте. В процессе выполнения второго прохода то же самое произойдет со следующим по величине элементом и т.д. Поэтому при последующих проходах можно уменьшать длину просмотра массива, что существенно сократит общее время выполнения

алгоритма. Если объединить этот метод оптимизации с проверкой признака завершения сортировки, то получится алгоритм обменной сортировки с признаком завершения.

4.2 Задание к лабораторной работе

Решить задачу, соответствующую варианту, выбранному по таблице 4.1. Построить блок-схему программы и организовать ввод данных, если это необходимо, с помощью генератора случайных чисел.

Таблица 4.1 – Варианты заданий

Вариант	Задание
1	Дана целочисленная прямоугольная матрица A(8,8). Методом простой сортировки расположить элементы в ее строках по возрастанию.
2	Дана целочисленная прямоугольная матрица B(10,10). Методом пузырьковой сортировки без оптимизации расположить элементы в ее столбцах по возрастанию.
3	Дана целочисленная матрица K(6,8). Методом пузырьковой сортировки с оптимизацией по времени выполнения каждого прохода расположить элементы по убыванию в столбцах.
4	Дана целочисленная матрица B(10,7). Методом пузырьковой сортировки с оптимизацией по времени выполнения каждого прохода расположить элементы по возрастанию в строках.
5	Дана целочисленная матрица A(8,7). Переписать все положительные элементы в вектор. Методом пузырьковой сортировки без оптимизации расположить элементы по возрастанию.
6	Переписать все положительные элементы матрицы X(10,10) в вектор Y. Используя метод пузырьковой сортировки с оптимизацией по количеству проходов, расположить по возрастанию.
7	Переписать все ненулевые элементы матрицы A(6,5) в вектор B. Используя метод пузырьковой сортировки с оптимизацией по количеству проходов, расположить по убыванию.
8	Дана целочисленная прямоугольная матрица C(9,9). Методом простой сортировки расположить элементы в ее столбцах по убыванию.
9	Переписать все ненулевые элементы матрицы A(8,5) в вектор B. Используя метод обменной сортировки с признаком завершения, расположить по убыванию.
10	Дана целочисленная матрица A(6,9). Переписать все отрицательные элементы в вектор. Методом пузырьковой сортировки без оптимизации расположить элементы по убыванию.

11	Дана целочисленная матрица P(7,5). Методом пузырьковой сортировки с оптимизацией по времени выполнения каждого прохода расположить элементы по убыванию в строках.
12	Дана целочисленная матрица A(5,9). Переписать все элементы в вектор по столбцам. Методом пузырьковой сортировки без оптимизации расположить элементы по возрастанию.
13	Дана целочисленная матрица C(6,7). Переписать все элементы в вектор по строкам. Методом обменной сортировки с признаком завершения расположить элементы по возрастанию.
14	Дана целочисленная матрица X(6,6). Переписать все элементы в вектор по столбцам. Методом обменной сортировки с признаком завершения расположить элементы по убыванию.
15	Дана целочисленная прямоугольная матрица A(10,10). Методом простой сортировки расположить элементы в ее столбцах по возрастанию.

4.3 Контрольные вопросы

4.3.1 Перечислите известные Вам методы сортировки.

4.3.2 В чем заключается суть метода сортировки простым выбором?

4.3.3 Приведите блок-схему метода сортировки простым выбором.

4.3.4 Перечислите отличительные особенности метода пузырьковой сортировки?

4.3.5 В чем смысл оптимизации метода пузырьковой сортировки?

4.3.6 Как можно уменьшить количество проходов сортировки при использовании метода пузырьковой сортировки?

4.3.7 С какой целью используется признак отсутствия перестановок при оптимизации метода пузырьковой сортировки?

4.3.8 В чем заключается суть оптимизации метода пузырьковой сортировки по времени выполнения каждого прохода?

4.3.9 Приведите блок-схему алгоритма оптимизации метода пузырьковой сортировки по времени выполнения каждого прохода.

4.3.10 Приведите пример алгоритма обменной сортировки с признаком завершения.

5 Лабораторная работа. Обработка символьных данных

Цель: получить практические навыки обработки символьной информации – строк.

5.1 Общие сведения

Строка представляет собой массив значений типа char, завершающийся нулевым байтом ‘\0’, что следует учитывать при объявлении строки, т.е.

указывать не N, а N+1 элемент. При инициализации строк используются традиционные методы объявления.

Для определения константы, равной длине инициализированной строковой переменной, можно воспользоваться функцией sizeof().

Для ввода символьных переменных и строк в С предназначены функции scanf() (ввод до первого пробельного символа) или gets() из библиотеки <stdio.h>. С++ дополнительно предоставляет пользователю две функции cin,get и cin.getline из библиотеки <iostream.h>. Для вывода в этих же библиотеках существуют аналогичные функции.

При работе со строками чаще всего используются функции библиотеки <string.h>, в частности:

а) *склеивание* – последовательное объединение нескольких строк:

```
strcat (str1, str2);
```

б) *копирование* строк:

```
strcpy(str1, str2);
```

в) *сравнение* строк:

```
strcmp(str1,str2);
```

г) *длина* строки:

```
lenth=strlen(str1);
```

д) *преобразование строчных* символов в прописные:

```
strlwr(str1);
```

е) *преобразование прописных* символов в строчные:

```
strupr(str1);
```

ж) *заполнение* строки некоторым *символом*:

```
strset(str1,'символ');
```

и) получить *код символа*:

```
n=int(a);
```

5.2 Задания к лабораторной работе

Решить задачу, соответствующую варианту, выбранному по таблице 5.1. Построить блок-схему программы и организовать ввод данных пользователем.

Таблица 5.1 – Варианты заданий

Вариант	Задание
1	Удвоить в строке s каждое вхождение буквы Z.
2	Заменить в строке s каждую первую букву слов, начинающихся с гласной буквы на прописную.
3	Определяет, сколько во введенной строке s слов, состоящих не более чем из четырех букв.
4	Вывести на экран введенное предложение, меняя местами каждые два соседних слова.
5	Вывести на экран все слова предложения, начинающиеся с гласных букв.

6	Из строки удалить среднюю букву, если длина строки нечетная, иначе – удалить две средние буквы. Средней считается буква, размещенная строго по центру строки.
7	Определить позицию начала в строке s слова с номером n.
8	Определить длину слова с номером n в строке s.
9	Подсчитать количество букв в слове с номером n строки s.
10	Вывести предложение на экран, записав все его слова в обратном порядке.
11	Заменить в строке s все вхождения подстроки str1 на подстроку str2.
12	Удалить из строки s подстроку s1, начиная с позиции n, длиной l.
13	Вставить в строку s подстроку s1, начиная с позиции n.
14	Подсчитать количество слов в строке s.
15	Скопировать подстроку s в строку s1 n раз.

5.3 Контрольные вопросы

5.3.1 Что представляет собой строка?

5.3.2 Как объявляются строковые переменные?

5.3.3 Назовите основные особенности строк.

5.3.4 Как можно определить длина инициализированной строковой переменной?

5.3.5 Как осуществляется ввод строк?

5.3.6 Приведите пример ввода строки фиксированной длины с использованием библиотеки <stdio.h>.

5.3.7 Какие функции предлагаются для вывода строк?

5.3.8 Приведите пример построчного ввода символьных данных.

5.3.9 Приведите пример посимвольного ввода строки.

5.3.10 Какие основные функции предусмотрены для работы со строками?

6 Лабораторная работа. Работа со структурами и объединениями

Цель: получить практические навыки работы с такими сложными типами данных в C++, как структуры и объединения.

6.1 Общие сведения

Структура (*struct*) состоит из фиксированного числа компонентов (элементов) разных типов. Описание типа struct имеет следующий вид:

```
struct  имя_структуры
{
тип1 имя_элемента1;
…
```

типN имя_элементаN; }
имя_переменной_типа_структуры;

Для обращения к элементу структуры нужно указать имя переменной и имя элемента структуры, разделив их точкой. К элементу структуры применима операция присваивания.

Объединения (*union*) в языке C++ отличаются от структур способом хранения информации. В каждый момент времени объединение хранит значение только одного элемента. Память распределяется для хранения наибольшего элемента объединения. Описание типа union имеет вид:

```
union   имя_объединения
{
тип1 имя_элемента1;
…
типN имя_элементаN;
} имя_переменной_типа_объединения;
```

Обращаются к элементу объединения аналогично обращению к элементу структуры. Очевидно, что использование объединений в программах позволяет экономить память компьютера. Еще большей экономии памяти в программах на языке C++ можно достичь, если использовать *анонимные* объединения. В анонимных объединениях нет имени, переменная объединения не объявляется:

```
union {
тип1   имя_элемента1;
…
типN   имя  элементаN;
} ;
```

К элементам анонимного объединения обращаются по имени (без точки), как к обычным переменным в программе.

6.2 Задание к лабораторной работе

Решить задачу, соответствующую варианту, выбранному по таблице 6.1. Построить блок-схему программы. Организовать ввод и вывод данных структуры (не менее 10 записей). Используя поля созданной структуры, выполнить предложенную выборку.

Таблица 6.1 – Варианты заданий

Вариант	Задание
1	Дана структура, в которой хранятся данные о полетах самолетов: название пункта назначения рейса, номер рейса, тип самолета. Вывести на экран номера рейсов и типы самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры. Если таких рейсов нет, выдать не дисплей соответствующее сообщение.

2	Имеются данные о работающих в фирме: фамилия, имя, отчество, адрес (улица, дом, квартира) и дата поступления на работу (месяц, год). Вывести на экран данные только тех из них, кто на сегодняшний день проработал уже не менее 5 лет.
3	Имеются данные о работающих на фабрике: фамилия, имя, отчество, адрес (улица, дом, квартира) и дата поступления на работу (месяц, год). Определить, есть ли в списке человек с указанной пользователем фамилией. Если есть, то вывести его адрес.
4	Имеются данные о клиентах пункта проката: фамилия, имя отчество, адрес (улица, дом, квартира) и что взял (только один предмет). Вывести на экран данные только тех из них, кто взял указанный предмет.
5	Даны сведения: фамилия, имя, знак зодиака, дата рождения. Вывести информацию о людях, родившихся в месяц, название которого введено с клавиатуры. Определить, кто родился в один год.
6	Имеются данные о расписании поездов: номер поезда, маршрут, время прибытия на станцию, время отправления (часы, минуты). Будем считать, что все поезда приходят каждый день. По данному времени определить, какие из поездов стоят сейчас на станции (время вводить с клавиатуры).
7	Дана информация о репертуаре кинотеатров: кинотеатр; фильм; жанр; период показа; время. Вывести на экран список фильмов, показанных в одном кинотеатре.
8	Дана структура данных о сотрудниках предприятия: фамилия, имя, адрес (улица, дом, квартира), отдел. Вывести фамилии сотрудников данного отдела, живущих на указанной улице.
9	Дана структура, в которой хранятся сведения о товарах: название товара, название магазина, в котором продается товар, стоимость товара в тенге. Вывести на экран информацию о товаре, название которого введено с клавиатуры. Если таких товаров нет, выдать соответствующее сообщение.
10	Имеются данные о студентах группы: фамилия, имя, отчество, адрес – улица, дом, квартира, домашний телефон. Вывести на экран фамилию, имя и адрес тех студентов, до кого нельзя дозвониться.
11	Дана структура, в которой хранятся сведения о плательщиках и получателях: расчетный счет плательщика, имя плательщика, расчетный счет получателя, имя получателя, перечисляемая сумма. Вывести на экран информацию о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры. Если такого расчетного счета нет, выдать соответствующее сообщение.

12	Даны сведения о собаках, участвующих в выставке: порода; пол; кличка; возраст; владелец. Вывести на экран информацию об участниках одного возраста.
13	Дана структура, в которой хранятся сведения о книгах в библиотеке: автор книги; название; издательство; год выпуска; количество. Определить, сколько книг указанного автора имеется в наличии.
14	Дана структура, в которой хранятся сведения о спортсменах: ФИО; гражданство; вид спорт; клуб (команда); дата рождения. Определить, кто из спортсменов выступает за одну команду.
15	Дана информация об автомобилях, имеющихся в автосалоне: модель; год выпуска; технические характеристики; техническое состояние; стоимость. Вывести на экран список автомобилей одного года выпуска.

6.3 Контрольные вопросы

6.3.1 Что представляет собой структура?

6.3.2 Чем отличается от структуры объединение?

6.3.3 Как описываются элементы структуры?

6.3.4 Как организуется обращение к элементам структуры и объединения?

6.3.5 Что понимается под анонимным объединением?

6.3.6 Как определить объем памяти, отводимой под структуру?

6.3.7 Сколько элементов можно хранить в структуре? В объединении?

6.3.8 Что понимается под структурным тэгом?

6.3.9 Как определяется тип структуры?

6.3.10 Что изменится, если в разработанной в лабораторной работе программе структуру заменить на объединение?

7 Лабораторная работа. Файлы и работа с ними

Цель: получить практические навыки работы с файлами в C++.

7.1 Общие сведения

Файл представляет собой последовательность элементов одного и того же типа, имеющих общее имя. Число элементов (длина файла) не ограничено. Файлы создаются на магнитной ленте, на магнитном диске, на других внешних устройствах, а также в оперативной памяти компьютера.

Для того чтобы программа на языке C++ могла работать с файлами, необходимо в начало программы включить заголовочный файл <fstream.h>.

а) запись данных в файл возможна в двух режимах:

1) создание нового файла (перезапись существующего, ранее созданного файла) с помощью оператора:

ofstream имя_потока ("имя_файла.расширение");

2) добавление данных в существующий файл:

ofstream out_file ("имя_файла. расширение", ios::app);

Вывод в файл осуществляется с помощью оператора вставки (<<).

Для закрытия файла используется функция close().

б) ввод (чтение) данных из файла. Для открытия файла в режиме ввода используется оператор:

ifstream in_file("имя_файла.расширение");

Ввод из файла осуществляется с помощью оператора извлечения (>>).

Иногда требуется записывать в файл и читать из файла не символьные строки, а сложные типы данных, такие, как массивы и структуры. Для этих целей в С++ используются функции write (запись) и read (чтение).

7.2 Задание к лабораторной работе

Решить задачу, соответствующую варианту, выбранному по таблице 7.1. Построить блок-схему программы. Организовать ввод/вывод данных с использованием файлов.

Таблица 7.1 – Варианты заданий

Вариант	Задание
1	Скопировать из одного файла в другой только указанные пользователем символы и посчитать их общее количество.
2	Считать текст из файла и вывести на экран только цитаты, то есть предложения, заключенные в кавычки.
3	Считать текст из файла и вывести его на экран, меняя местами каждые два соседних слова.
4	Считать текст из файла и вывести на экран только предложения, состоящие из заданного количества слов.
5	Считать текст из файла и вывести на экран все его предложения в обратном порядке.
6	Сформировать и вывести в файл квадратную матрицу A(10,10), в которой все нулевые элементы размещены квадратами 2x2 в шахматном порядке.
7	Сформировать и вывести в файл квадратную матрицу C(12,12), в которой значения элементов, размещенных на местах с четными индексами строк и столбцов, равны квадрату индекса строки.
8	Сформировать и вывести в файл квадратную матрицу A(6,6), в которой значение каждого элемента определяется как сумма его индексов.
9	Дан файл, в котором хранится расписание полетов: номер рейса, тип самолета, дни и время вылета рейса, название пунктов вылета-назначения рейса, время в пути. Вывести на экран расписание, отсортированное по номерам рейсов.

10	Сформировать и вывести в файл квадратную матрицу B (7,7), все ненулевые элементы которой размещены в шахматном порядке, начиная с 1-го элемента 1-й строки.
11	Имеются сведения о результатах сессии студентов одного факультета, которые хранятся в структуре. Вывести в файл информацию о результатах сессии студентов указанной группы.
12	Имеются сведения о результатах сессии студентов одной группы, которые хранятся в файле. Вывести на экран информацию о результатах сдачи указанного экзамена.
13	Дана квадратная матрица C(8,8). Транспонировать ее и вывести результат в файл.
14	Дан файл, в котором через пробел записаны натуральные четырехзначные числа. Вывести на экран суммы цифр каждого числа.
15	Дан файл с учебным расписанием. Вывести на экран количество лекционных, практических и лабораторных занятий по указанной дисциплине.

7.3 Контрольные вопросы

7.3.1 Что представляют собой файлы?
7.3.2 Какие типы файлов Вам известны?
7.3.3 Какие библиотечные функции применяются при работе с файлами?
7.3.4 Из каких шагов складывается работа с файлами?
7.3.5 Какие операторы используются при работе с файлами?
7.3.6 Перечислите режимы записи данных в файл?
7.3.7 Как добавить данные в файл?
7.3.8 Приведите пример чтения данных из файла.
7.3.9 Приведите пример записи массива в файл.
7.3.10 Каковы особенности записи массива в файл?

8 Лабораторная работа. Функции. Рекурсия

Цель: получить практические навыки использования функций в C++.

8.1 Общие сведения

Кроме стандартных функций, размещенных в заголовочных файлах, язык C++ позволяет формировать пользовательские функции. Эти функции целесообразно создавать, если при решении задач возникает необходимость проводить вычисления многократно по одним и тем же алгоритмам. Применение функций позволяет разделить программу на простые, легко контролируемые части. При использовании функций необходимо различать описание функции и оператор вызова функции.

Структура функции похожа на структуру программы main. Описание функции содержит заголовок функции, объявления переменных и операторы:

тип_функции имя_функции (список формальных_параметров)

{

объявления переменных;

оператор1;

…;

операторы; }

Здесь: тип_функции — тип возвращаемого в основную программу результата;

имя_функции — уникальное имя, соответствующее по смыслу операции, которую выполняет функция (например, max — определение максимального из двух чисел);

список_формальных_параметров — перечень формальных параметров и их типов.

Формальные параметры — это наименования переменных, через которые передается информация из основной программы в функцию.

Для вызова функции достаточно указать ее имя со списком фактических параметров в любом выражении вызывающей программы:

имя_функции (список_фактических_параметров);

Фактические параметры — это наименования переменных, значения которых при обращении к функции, присваиваются соответствующим формальным параметрам.

Для возвращения вычисленного значения в основную программу в функциях используется оператор return(результат);

Если описание функции размещается в программе после обращения к ней, в начало программы нужно поместить прототип функции. Прототип функции содержит информацию об имени функции, типе возвращаемого значения, количестве и типе формальных параметров, например:

float max(float, float);

Кроме переменных, в качестве формальных параметров в функциях могут выступать и массивы.

Рекурсия - вычислительный процесс, направленный на решение определенной задачи таким образом, что само решение использует этот же процесс, решающий аналогичную подзадачу. В программировании под рекурсией понимают такую реализацию, в которой подпрограмма использует в своем теле вызов самой себя. Такие вызовы называют рекурсивными. Когда функция А в своем теле вызывает только одну рекурсивную функцию (саму себя), то говорят о простой рекурсии. Под косвенной рекурсией понимают явление, когда рекурсивные функции вызывают друг друга (например, функция А вызывает В, а функция В вызывает А).

Рекурсивные алгоритмы сложнее отлаживать, но порой они позволяют очень гибко и красиво решить задачу. Известно, что любой рекурсивный алгоритм можно заменить нерекурсивным, но это будет стоить больше

времени на реализацию. Главное, чтобы рекурсивная функция не вызывала себя бесконечно, иначе программа не будет работать. Поэтому при реализации рекурсивных алгоритмов необходимо уделять внимание тому, чтобы алгоритм был конечным, т.е. выполнение рекурсивной функции должно когда-нибудь завершиться.

8.2 Задание к лабораторной работе

Решить задачу с использованием рекурсивной функции (таблица 8.1). Построить блок-схему программы. Организовать ввод и вывод данных.

Таблица 8.1 – Варианты заданий

Вариант	Задание
1	$A = \frac{C!}{(C-n)!}(C+n)!$
2	$Z = \frac{(n+1)!}{(m-n)!(m+n)!}$
3	$Y = \frac{(n+1)!(n+2)!}{(kn)!k!}$
4	$y = \sum_{n=1}^{n=20} \frac{x^n}{(2n-1)!}$
5	$a^x = 1 + \sum_{n=1}^{n=15} \frac{(x \ln a)^n}{n!}$
6	$S = \sum_{n=1}^{n=10} \frac{\sin x^{n!}}{2n!}$
7	$S = \sum_{n=1}^{n=15} \frac{n!}{(n!+1)-(n+2)!}$
8	$S = \frac{2}{1!} + \frac{2^2}{2!} + \ldots + \frac{2^n}{n!}$
9	$a^x = 1 + \frac{x \ln a}{1!} + \frac{(x \ln a)^2}{2!} + \ldots + \frac{(x \ln a)^n}{n!}$
10	$P = \frac{2}{1!} * \frac{2^2}{2!} * \ldots * \frac{2^n}{n!}$
11	$S = \frac{2}{2!} + \frac{2^2}{4!} + \ldots + \frac{2^n}{2n!}$
12	$z = 1 + \frac{x}{(2n-1)!} - \frac{x^2}{(2n-3)!} + \ldots - \frac{x^n}{1!}$

13	$y = 1 + \frac{x}{1!} + \frac{x^2}{3!} + ... + \frac{x^n}{(2n-1)!}$
14	$z = x - \frac{1}{1!} + \frac{1}{3!} - ... + \frac{1}{(2n-1)!}$
15	$S = \frac{x}{n!} + \frac{x^2}{(n-1)!} + \ldots + \frac{x^n}{1!}$

8.3 Контрольные вопросы

8.3.1 Что представляет собой функция?

8.3.2 Опишите структуру функции?

8.3.3 По каким правилам оформляются заголовки функций?

8.3.4 В чем разница между формальными и фактическими параметрами?

8.3.5 Как осуществляется объявление локальных и глобальных переменных?

8.3.6 С какой целью в программах используются прототипы функций?

8.3.7 Что такое перегрузка функций?

8.3.8 Что такое рекурсия? В чем отличие между простой и косвенной рекурсией?

8.3.9 Назовите достоинства и недостатки рекурсивных функций.

8.3.10 Приведите примеры использования рекурсивных функций.

Список литературы

1 Страуструп Б. Язык программирования C++. – М., 2012.

2 Павловская Т.А. C/C++. Структурное программирование. – СПб., 2010.

3 Немцова Т.И. Программирование на языке высокого уровня. Программирование на языке C++. М.: «Форум», 2012.

4 Ашарина И.В. Основы программирования на языках С и С++. - М., Горячая линия-Телеком, 2012.

5 Аляев Ю.А., Козлов О.А. Алгоритмизация и языки программирования Pascal, C++, VisualBasic: Учебно - справочное пособие. – М.: Финансы и статистика, 2004.

6 Культин Н. C/C++ в задачах и примерах. – СПб.: Питер, 2011.

Содержание

Сводный план 2014 г., поз. 117

Наталья Валерьевна Сябина
Лариса Николаевна Рудакова

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Методические указания по выполнению лабораторных работ
для студентов специальности 5В070200

Редактор Л.Т. Сластихина
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать __. __. __.
Тираж *100* экз.
Объем *1.7* уч.-изд. л.

Формат 60х84 1/16
Бумага типографская №1
Заказ _____. Цена 850 тг.

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, ул. Байтурсынова, 126