МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических работ по дисциплине

«СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

для студентов направления подготовки
44.03.05 Педагогическое образование (с двумя профилями подготовки)
направленность (профиль):
«Информатика и математика»

Ставрополь, 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
СТРУКТУРА И СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ РАБОТ 4
Практическая работа 1 Знакомство с интегрированной средой DevC++. Составление программ линейной структуры 4
Практическая работа 2. C++. Составление программ разветвляющейся структуры. Оператор IF 6
Практическая работа 3. C++. Составление программ разветвлённой структуры. Оператор SWITCH 6
Практическая работа 4. C++. Составление программ циклической структуры. Оператор for 6
Практическая работа 5. C++. Составление программ циклической структуры. Обработка последовательностей. Операторы while, do… while 6
Практическая работа 6. C++. Обработка одномерных массивов 8
Практическая работа 7. C++. Обработка двумерных массивов 9
Практическая работа 8. Классы и объекты в C++ 10
Практическая работа 9. C++. Наследование и виртуальные функции 10
Практическая работа 10. C++. Иерархия объектов и группа 11
Практическая работа 11. C++. Итераторы 11
Практическая работа 12. C++. Обработка событий 11
Практическая работа 13. C++. Перегрузка операций 11
Практическая работа 14. C++. Шаблоны функций и классов 11
Практическая работа 15. C++. Потоковые классы 12
Практическая работа 16. C++. Стандартная библиотека шаблонов 12
УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 13

ВВЕДЕНИЕ

1. Цель и задачи освоения дисциплины (модуля)

Цель дисциплины: формирование набора профессиональных и профессионально-профильных компетенций будущего бакалавра (специалиста) по соответствующему направлению подготовки.

Задачи дисциплины:

– раскрыть понятийный аппарат фундаментального и прикладного аспектов дисциплины;

– освоить начала программирования на языке C++;

– сформировать навыки работы в среде программирования Dev C++, навыки тестирования и отладки программ;

– изучить основы классической теории объектно-ориентированного программирования;

– овладеть средствами объектно-ориентированного программирования языка C++, средствами стандартной библиотеки STL для решения практических задач получения, хранения, обработки и передачи информации.

В результате освоения дисциплины студенты должны:

Знать:

– методы отбора учебного содержания в соответствии с дидактическими целями и возрастными особенностями обучающихся

– методы отбора вариативного содержания с учетом взаимосвязи урочной и внеурочной форм обучения информатике и ИКТ в профессиональной деятельности учителя;

– основные методы математической обработки информации;

– основные приемы алгоритмизации и программирования на ЯВУ;

– методы структурного и объектно-ориентированного программирования;

– принципы автономной отладки и тестирования программ.

Уметь:

– осуществлять отбор учебного содержания в соответствии с дидактическими целями и возрастными особенностями обучающихся

– производить отбор вариативного содержания с учетом взаимосвязи урочной и внеурочной форм обучения информатике и ИКТ в профессиональной деятельности учителя информатики;

– применять основные методы математической обработки информации, основные приемы алгоритмизации и программирования на ЯВУ;

–использовать методы структурного и объектно-ориентированного программирования; принципы автономной отладки и тестирования программ;

Владеть:

– современными компьютерными технологиями для решения практических задач получения, хранения, обработки и передачи информации;

– навыками программирования в современных средах для решения практических задач получения, хранения, обработки и передачи информации

– навыками тестирования и отладки программ; оформления программной документации.

Наименование практических работ

№	Наименование тем дисциплины, их краткое содержание	Объем часов
7 семестр		
1	Практическая работа № 1 Знакомство с интеграционной средой Dev C++. Составление программ линейной структуры	2
2	Практическая работа № 2 C++. Составление программ разветвляющейся структуры. Оператор if.	2
3	Практическая работа № 3 C++. Составление программ разветвляющейся структуры. Оператор switch	2
4	Практическая работа № 4 C++. Составление программ циклической структуры. Оператор for	2
5	Практическая работа № 5 C++. Составление программ циклической структуры. Обработка последовательностей. Операторы while и do ... while	2
6	Практическая работа № 6 C++. Обработка одномерных массивов	2
7	Практическая работа № 7 C++. Обработка двумерных массивов	2
8	Практическая работа № 8 Классы и объекты в C+	2
9	Практическая работа № 9 C++. Наследование и виртуальные функции	2
10	Практическая работа № 10 C++. Иерархия объектов и группа	2
11	Практическая работа № 11 C++. Итераторы	2
12	Практическая работа № 12 C++. Обработка событий	2
13	Практическая работа № 13 C++. Перегрузка операций	2
14	Практическая работа № 14 C++. Шаблоны функций и классов	2
Итого за 7 семестр		**28**
Итого		**28**

СТРУКТУРА И СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Практическая работа 1 **Знакомство с интегрированной средой DevC++. Составление программ линейной структуры**

Цель работы: приобретение навыков работы в интегрированной среде DevC++; освоение процесса ввода, отладки и выполнения программ; изучение основных математических операций и стандартных функций языка C/C++; освоение функций ввода и вывода данных, приобретение навыков составления алгоритмов и программ линейной структуры.

Задания:

1. Знакомство с интерфейсом DevC++:

– создайте папку на flash-носителе для хранения исходных текстов программ на языке C++.

– запустите интегрированную среду DevC++.

– ознакомьтесь с интегрированной средой, описание которой приведено в теоретической части работы.

– откройте окно редактирования программ командой File\New source file или щелчком по одноименному инструменту (рис. 10.3.1.2). В окне редактирования появляется «заготовка» для программ на языке C++.

2.Запись математических выражений на языке C/C++

Запишите на языке C/C++ выражения (программу писать не нужно!!!):

а) $\frac{y \cdot e^{\sqrt{|w|}} \cdot tgx \cdot w}{1.85w + \sqrt[3]{y^2}}$ б) $a^6 + \cos^4 x \frac{1}{\sqrt[3]{j}}$

в) $T = \frac{1}{\cos x} + \ln\left|tg\left(\frac{x}{2}\right)\right|$ г) $F = 0{,}23x\sqrt{\frac{1+\left|\cos\ x^3\right|}{ab}}$

3. Программирование линейной структуры

Составьте блок-схему и напишите программу линейной структуры для решения следующих задач:

а) С клавиатуры вводятся: целые числа A и B выражающие объем информации в килобайтах. Перевести A в байты, B в мегабайты.

б) С клавиатуры вводится сторона равностороннего треугольника A. Вычислить радиус вписанной r и описанной R окружностей этого треугольника.

в) С клавиатуры вводятся: скорость лодки в стоячей воде V, скорость течения реки Vr, время движения по озеру T1, а против течения – T2. Вычислить путь, пройденный лодкой.

4. Операции целочисленного деления / и %

Вычислите:

15 % 2 =

34 / 8 + 33 % 6 =

36 / 9 =

Определите значение переменной a после выполнения алгоритма:

```
a:= 15
b:=47 % a
b:=b*10
a:=b / 4 + 5
```

```
a:=28
b:=a / 10-3
a:=b+10
```

```
a:=5
b:=a *3
b:=b / 2+a
a:=3*(b %a+8)
```

```
a:=568
b:=a /100
c:=(b /10)%10+a
a:=b+c+a %10
```

Практическая работа 2. C++. **Составление программ разветвляющейся структуры. Оператор IF**

Цель работы: приобретение навыков разработки алгоритмов и программ разветвляющейся структуры с использованием условного оператора.

Задания:

1.Составьте алгоритм и напишите программу разветвленной структуры для решения следующих задач:

а) С клавиатуры вводятся числа K, L, M. Подсчитать количество четных чисел.

б) С клавиатуры вводятся стороны двух прямоугольников A, B и X, Y. Определить, равны ли площади прямоугольников.

в) Найти действительные корни уравнения: AX2 + BX + C = 0. Если корней нет, то выдать соответствующее сообщение. Учесть случай, когда A=0.

Практическая работа 3. C++**. Составление программ разветвлённой структуры. Оператор SWITCH**

Цель работы: приобретение навыков разработки алгоритмов и программ разветвляющейся структуры с использованием оператора множественного выбора.

Задания:

1. Откомпилируйте задачи, рассмотренные на лекции.

2. Составьте блок-схемы и напишите программы разветвленной структуры для решения следующих:

а) С клавиатуры вводится оценка, полученная студентом – Oc. Выдать сообщение ‘Умница’ , если Oc=5, ‘Хорошо’ – Oc=4, ‘Лентяй’ – Oc=3 или Oc=2.

б) С клавиатуры вводится первая буква названия падежа (И – именительный, Р – родительный, Д – дательный, В – винительный, Т – творительный, П – предложный). Вывести слово студент в заданном падеже единственного числа.

в) С клавиатуры вводится целое число N[1; 5]. Вывести фразу «Мы сдали N экзаменов», учитывая, что при некоторых значениях N слово «экзаменов» необходимо заменить словом «экзамен» или «экзамена».

Практическая работа 4. **C++. Составление программ циклической структуры. Оператор for**

Цель работы: приобретение практических навыков в составлении алгоритмов и программ циклической структуры.

Задания:

1. Составьте блок-схему и напишите программу перевода фунтов в кг от 1 фунта до 200 с шагом 20 фунтов. 1 фунт = 400 г.

2. Составьте блок-схему и напишите программу циклической структуры, используя оператор со счётчиком, для решения следующей задачи.

Дано целое число N (N□[1;20]). Вычислить значение выражения:

1*1 – 1*2 + 1*3 – …+/–1*N .

Практическая работа 5. **C++. Составление программ циклической структуры. Обработка последовательностей. Операторы while, do… while**

Цель работы: приобретение практических навыков в составлении алгоритмов и программ циклической структуры.

Теоретическая часть

Операторы цикла часто используются для обработки числовых последовательностей. Алгоритм для обработки последовательнойтей относится к одному из двух типов: поиск, проверка условий.

Для последовательностей характерно, что в каждый момент времени нам доступен только один элемент последовательности, а количество элементов последовательности пользователь может определять вводом с клавиатуры. Поэтому все алгоритмы обработки последовательностей строятся с учетом однократного последовательного просмотра.

Пример 1. Даны N чисел. Найти количество отрицательных чисел, используя цикл for.

```
# include <iostream.h>
# include <math.h>
main()
{ float a;  int kol, i, n;     // объявление переменных
 cout <<"\n  Введите количество чисел";
 cin  >> N;                  // ввод значения N
 for (kol=0, i = 1;  i<= n;  i++)//заголовок цикла, перв.кол-во чисел kol=0
          {cout <<"Введите число";   cin  >> a;
          if   (a<0)   kol ++; //прибавление единицы к kol, если a<0
          }//возвращение на начало цикла с увеличением счетчика на шаг
cout <<  "\n Количество  всех отрицательных  чисел="<< kol;
return 0;}
```

Пример 2. Даны N чисел. Найти наибольшее среди них, используя цикл for.

```
…    // объявление переменных и ввод значения N
cout <<"Введите число";
cin  >> a;
for (max = a, i = 2;  i<= n;  i++)//первоначальное значение max
                  {cout <<"Введите число";
cin  >> a;
          if   (a >max)   max=a;  //сравнение числа с максимальным
                  }
cout<< "\n  Наибольшее  из чисел =";  max
```

Пример 3. Даны N чисел. Найти сумму положительных чисел, используя цикл for.

```
…    // объявление переменных и ввод значения n
for (S=0, i = 1; i<= n; i++)
          {cout <<"Введите число";   cin  >> a;
          if   (a>0)   S +=a; //прибавление числа к сумме, если a<0
          }//возвращение на начало цикла с увеличением счетчика на шаг
cout <<  "\n Сумма  всех отрицательных  чисел="<< S; }
```

Циклы с неизвестным количеством повторений

Формат записи цикла с предусловием:

```
while (условие) тело цикла;
```

Формат записи цикла с постусловием:

```
do
{тело цикла}
while (условие);
```

Циклы с предусловием и постусловием продолжаются до тех пор, пока условие истинно. В отличие от цикла с предусловием, цикл с постусловием выполнится хотя бы один раз.

Пример 4. Изменим условие в примере 3: Дана последовательность чисел N и величина K, введенная с клавиатуры. Определить, сколько чисел надо ввести, чтобы их сумма превысила величину K. Такую программу можно реализовать с помощью циклов предусловия или постусловия:

```
 // цикл с пред. условием
 #include<iostream.h>
 main()
```

```
 { int K,S, i;
 cout << “Введите К\n”;   cin>>K;
 S=0;
 i= 0;
while ( S<K)
        { cout <<”Введите число”;
cin  >> a;
i++ ;
S+=a;}
        cout <<“Кол–во введенных чисел=“ << i<< “их сумма=“<<S;
  return 0; }
//цикл с пост. условием
#include<iostream.h>
 main()
 { int K,S, i;
 cout << “Введите К\n”;   cin>>K;
 S=0;  i= 0;
do
{ cout <<”Введите число”;
cin  >> a;
i++ ;
S+=a;}
while ( S<K) ;
  cout <<“Кол–во введенных чисел=“ << i<< “их сумма=“<<S;
  return 0;  }
```

Задания:

1. Построить блок-схемы к программам, приведенным в примерах 1-4. Проверить работу программ в среде DevC++..

Практическая работа 6. **C++. Обработка одномерных массивов**

Цель работы: приобретение практических навыков в составлении программ обработки одномерных массивов.

Задания:

1. Построить блок-схему программы ввода элементов массива и поиска максимального элемента.

```
//poiskmax.c  поиск максимального элемента в массиве.
 #include<stdio.h>
 #define n 10
 int i,max,k,a[n]; // размер массива n
 main() {
 for(i=0;i<n;i++) scanf("%d",&a[i]);
 max=a[0]; k=0;
 for(i=0;i<n;i++)
  { if(max<a[i])
                { k=i;
 max=a[i];
                }
 }
 printf("Максимальное число в массиве %d, его индекс %d\n",max,k);
 return 0; }
```

Проверить работу программы в среде DevC++.

2. Составьте блок-схемы и напишите программы обработки одномерных массивов для решения следующих задач:

а) Дан массив A[15]. Вычислить среднее арифметическое всех чисел массива с нечетными индексами и определить количество чисел равных нулю.

б) Дан массив B[9]. Заменить отрицательные элементы массива на 0.

Практическая работа 7. **C++. Обработка двумерных массивов**

Цель работы: приобретение практических навыков в составлении программ обработки двумерных массивов

Задания:

1. Построить блок-схему к приведенной программе и откомпилировать в среде DevC++.

Дана матрица A(3,4). Найти суммы элементов каждой строки и записать их в новый массив.

```
#include <iostream.h>
main ( )
{  const   int N=3, M=4;
   int A[N][M], sum [N] ;  // описание матрицы A  и вектора sum
   int  i, j;
  for ( i=0 ; i<N; i++)                    // ввод матрицы
          for ( j=0; j<M; j++)
                  {cout  << “Введите элемент  матрицы A=” <<endl;
                   cin  >> A[i] [j];        }
  cout <<“\n Вывод  матрицы A:” <<endl ;
 for (i=0; i<N; i++)     // вывод исходной  матрицы  по строкам
            { for (j=0; j<M; j++)   cout <<  A [i] [j] <<”  “;
              cout <<endl;          // переход на новую строку в матрице
            }
for ( i=0;  i<N;  i++)
          {sum [ i ]=0;                       // обнуление суммы в каждой строке
           for ( j=0;  j<M;  j++)       sum [i]+ =  A[i] [ j];
          }
cout <<endl;
 cout <<“Вывод массива сумм в столбец:<< endl;
      for (i=0; i<N; i++)  cout << sum [ i ]  << endl;
return 0; }
```

Применяя потоковый неформатный вывод элементов в примере 4, матрица выглядит несимметрично, если значения элементов разной длины. Чтобы элементы матрицы располагались один под другим, используют манипулятор setw (максимальная ширина поля вывода), который находится в заголовочном файле iomanip.h. Добавьте в программу строку с подключением этой и измените вывод матрицы следующим образом:

```
#include < iomanip.h >
for (j=0;   j<M;  j++)   cout <<  setw(4) <<A [i] [j] <<”  “
```

2, Построить блок-схему к приведенной программе и откомпилировать в среде DevC++.

Программа умножения матриц (двумерных массивов) по правилу матричного умножения:

$$c_{ij} = \sum_{k=1}^{l} a_{ik} \cdot b_{kj}$$

, где матрица a имеет размер n x l, матрица b имеет размер l x m, а матрица c - размер n x m.

```
// matrix.c умножение матриц
#include <stdio.h>
#define n 3
#define m 4
#define l 3
main(){
float a[n][l], b[l][m], c[n][m], s;
int i,j,k;
```

```
//ввод матрицы a
for(i=0; i<n; i++)
  for(j=0; j<l; j++){
printf("Введите a[%d][%d] ", i, j);
scanf("%f", a[i][j]);}
//ввод матрицы b
for(i=0; i<l; i++)
  for(j=0; j<m; j++){
printf("Введите b[%d][%d] ", i, j);
scanf("%f", b[i][j]);}
//вычисление и печать матрицы c
for(i=0; i<n; i++){
  for(j=0; j<m; j++){
for(k=0,s=0; k<l; k++){
s+=a [i] [k] * b [k] [j];
c [i] [j] = s;
printf("%10.4f",c[i][j]);}
}
  printf("\n");}
return 0;}
```

3. Составьте алгоритм и напишите программу обработки двумерных массивов для решения следующих задач:

а) Дана матрица P размером 2*3. Найти произведение элементов матрицы, которые по модулю меньше 10.

б) Определить максимальный элемент целочисленного массива C (3 *4). Четные элементы матрицы умножьте на максимальный элемент, а к нечетным элементам прибавьте максимальный элемент.

Практическая работа 8. **Классы и объекты в C++**

Задания:

1. Определить пользовательский класс в соответствии с вариантом задания.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных.
5. Определить указатель на компоненту-функцию.
6. Определить указатель на экземпляр класса.
7. Написать демонстрационную программу, в которой создаются и разрушаются объекты пользовательского класса и каждый вызов конструктора и деструктора сопровождается выдачей соответствующего сообщения (какой объект, какой конструктор или деструктор вызвал).
8. Показать в программе использование указателя на объект и указателя на компоненту-функцию.

Практическая работа 9. **C++. Наследование и виртуальные функции**

Задания:

1. Определить иерархию классов (в соответствии с вариантом).
2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.
3. Реализовать классы.
4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.
5. Сделать соответствующие методы не виртуальными и посмотреть, что будет.

6. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

Практическая работа 10. **C++. Иерархия объектов и группа**

Задания:

Дополнить иерархию классов предыдущей лабораторной работы классами “группа”.

Например, для предметной области ФАКУЛЬТЕТ можно предложить классы “факультет”, “студенческая группа”, “кафедра”. Рекомендуется создать абстрактный класс “подразделение”, который будет предком всех групп и абстрактный класс TObject, находящийся во главе всей иерархии.

2. Написать для класса-группы метод-итератор.

3. Написать процедуру или функции, которая выполняется для всех объектов, входящих в группу.

4. Написать демонстрационную программу, в которой создаются, показываются и разрушаются объекты-группы, а также демонстрируется использование итератора.

Практическая работа 11. **C++. Итераторы**

Задания:

1. Составить шаблон функции печати сформированного вами вектора для любых типов его элементов. Размеры вектора в функцию не передавать!

2. Реализовать требуемый алгоритм задания методами контейнера vector и алгоритмами STL.Если это не противоречит условию задачи, то заполнение вектора производить с помощью функций rand и generate.

Практическая работа 12. **C++. Обработка событий**

Задания:

1. Что собой представляет объект группа экранных форм?
2. Каким образом можно обратиться при работе приложения к конкретному методу или свойству одной из форм группы экранных форм?
3. Каким образом создаётся группа экранных форм?
4. Каким образом добавить новое свойство и новый метод в форму

Практическая работа 13. **C++. Перегрузка операций**

Задания:

1. Реализовать классы и операции (функции) для работы с объектами этих классов.
2. Записать полученную программу

Практическая работа 14. **C++. Шаблоны функций и классов**

Задания:

1.Создать шаблон заданного класса. Определить конструкторы, деструктор, перегруженную операцию присваивания (“=”) и операции, заданные в варианте задания.

2. Написать программу тестирования, в которой проверяется использование шаблона для стандартных типов данных.

3. Выполнить тестирование.

4. Определить пользовательский класс, который будет использоваться в качестве параметра шаблона. Определить в классе необходимые функции и перегруженные операции.

5. Написать программу тестирования, в которой проверяется использование шаблона для пользовательского типа.

6. Выполнить тестирование.

Практическая работа 15. C++. **Потоковые классы**

Задания:

1.Определить пользовательский тип данных (класс). Определить и реализовать в нем конструкторы, деструктор, операции присваивания, ввода и вывода для стандартных потоков.

2. Написать программу № 1 для создания объектов пользова-тельского класса (ввод исходной информации с клавиатуры с использованием перегруженной операции “>>) и сохранения их в потоке (файле). Предусмотреть в программе вывод сообщения о количестве сохраненных объектов и о длине полученного файла в байтах.

3. Выполнить тестирование программы.

4. Реализовать для вывода в поток свой манипулятор с параметрами.

5. Написать программу № 2 для чтения объектов из потока, сохранения их в массиве и просмотра массива. Для просмотра объектов использовать перегруженную для cout операцию << и свой манипулятор. Предусмотреть в программе вывод сообщения о количестве прочитанных объектов и байтов.

6. Выполнить программу для чтения из файла сохраненных предыдущей программой объектов и их просмотра.

7. Написать программу № 3 для добавления объектов в поток.

8. Выполнить программу, добавив в поток несколько объектов и просмотреть полученный файл.

9. Написать программу № 4 для удаления объектов из файла.

10. Выполнить программу, удалив из потока несколько объектов и просмотреть полученный файл.

11. Написать программу № 5 для корректировки (т.е. замены) записей в файле.

12. Выполнить программу и просмотреть полученный файл.

Практическая работа 16. **C++. Стандартная библиотека шаблонов**

Задания:

Написать и отладить три программы. Первая программа демонстрирует использование контейнерных классов для хранения встроенных типов данных.

Вторая программа демонстрирует использование контейнерных классов для хранения пользовательских типов данных.

Третья программа демонстрирует использование алгоритмов STL.

В программе № 1 выполнить следующее:

1. Создать объект-контейнер в соответствии с вариантом задания и заполнить его данными, тип которых определяется вариантом задания.

2. Просмотреть контейнер.

3. Изменить контейнер, удалив из него одни элементы и заменив другие.

4. Просмотреть контейнер, используя для доступа к его элементам итераторы.

5. Создать второй контейнер этого же класса и заполнить его данными того же типа, что и первый контейнер.

6. Изменить первый контейнер, удалив из него n элементов после заданного и добавив затем в него все элементы из второго контейнера.

7. Просмотреть первый и второй контейнеры.

В программе № 2 выполнить то же самое, но для данных пользовательского типа.

В программе № 3 выполнить следующее:

1. Создать контейнер, содержащий объекты пользовательского типа. Тип контейнера выбирается в соответствии с вариантом задания.

2. Отсортировать его по убыванию элементов.

3. Просмотреть контейнер.

4. Используя подходящий алгоритм, найти в контейнере элемент, удовлетворяющий заданному условию.

5. Переместить элементы, удовлетворяющие заданному условию в другой (предварительно пустой) контейнер. Тип второго контейнера определяется вариантом задания.

6. Просмотреть второй контейнер.

7. Отсортировать первый и второй контейнеры по возрастанию элементов.

8. Просмотреть их.

9. Получить третий контейнер путем слияния первых двух.

10. Просмотреть третий контейнер.

11. Подсчитать, сколько элементов, удовлетворяющих заданному условию, содержит третий контейнер.

12.Определить, есть ли в третьем контейнере элемент, удовлетворяющий заданному условию.

УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

1. Перечень основной литературы:

– Фридман, А.Л. Язык программирования Си++ Электронный ресурс : учебное пособие / А.Л. Фридман. - Язык программирования Си++,2021-01-23. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 218 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0017-5, экземпляров неограничено

– Агафонов, Е. Д. Прикладное программирование / Е.Д. Агафонов ; Г.В. Ващенко. - Красноярск : Сибирский федеральный университет, 2015. - 112 с. - ISBN 978-5-7638-3165-8, экземпляров неограничено

2. Перечень дополнительной литературы:

– Зюзьков, В.М.; Программирование Электронный ресурс : учебное пособие / В.М. Зюзьков. - Томск : Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013. - 186 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4332-0141-5, экземпляров неограничено

3. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

– Методические указания по выполнению практических работ по дисциплине «Современные языки программирования» (электронный ресурс).

– Методические рекомендации по организации проведению самостоятельной работы по дисциплине «Современные языки программирования» (электронный ресурс).

4. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

– http://el.ncfu.ru/ – система управления обучением ФГАОУ ВО СКФУ. Дистанционная поддержка дисциплины «Цифровая грамотность и обработка данных»

– http://www.intuit.ru – Интернет-Университет Компьютерных технологий.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для обучающихся по организации и проведению самостоятельной работы
по дисциплине

«СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

44.03.05 Педагогическое образование (с двумя профилями подготовки)
направленность (профиль):
«Информатика и математика»

Ставрополь, 2026

1. **Общие положения**

Самостоятельная работа – планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

− формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);

− написание докладов;

− подготовка к семинарам, практическим и лабораторным работам, их оформление;

− составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);

− выполнение учебно-исследовательских работ, проектная деятельность;

− подготовка практических разработок и рекомендаций по решению проблемной ситуации;

− выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;

− компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;

− выполнение курсовых работ (проектов) в рамках дисциплин;

− выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине **«Современные языки программирования»** направлена на формирование следующих **компетенций**:

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ОПК-8: Способен осуществлять	**ОПК-8.1** Применяет методы анализа педагогической ситуации,	Знает и применяет методы анализа педагогической

педагогическую деятельность на основе специальных научных знаний	профессиональной рефлексии на основе специальных научных знаний, в том числе в предметной области	ситуации, профессиональной рефлексии на основе специальных научных знаний, в том числе в предметной области
ПК-3: Способен формировать развивающую образовательную среду для достижения личностных, предметных и метапредметных результатов обучения средствами преподаваемых учебных предметов	**ПК-3.1** Владеет способами интеграции учебных предметов для организации развивающей учебной деятельности (исследовательской, проектной, групповой и др.) **ПК-3.2** Использует образовательный потенциал социокультурной среды региона в преподавании (предмета по профилю) в учебной и во внеурочной деятельности	Применяет знания и умения отбора учебного содержания в соответствии с дидактическими целями и возрастными особенностями обучающихся и вариативного содержания с учетом взаимосвязи урочной и внеурочной форм обучения информатике и ИКТ в профессиональной деятельности учителя.
ПК-10: Способен использовать математический аппарат, методологию программирования и современные компьютерные технологии для решения практических задач получения, хранения, обработки и передачи информации	**ПК-10.2** Применяет математический аппарат, методологию программирования и современные компьютерные технологии для решения практических задач получения, хранения, обработки и передачи информации	Понимает и применяет основные методы математической обработки информации, основные приемы алгоритмизации и программирования на ЯВУ; методы структурного и объектно-ориентированного программирования; принципы автономной отладки и тестирования программ Использует современные компьютерные технологии, методы и технологии программирования на современных ЯП ВУ для решения практических задач. Владеет навыками программирования в современных средах для решения практических задач получения, хранения, обработки и передачи информации

2. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование набора общенаучных, профессиональных и специальных компетенций будущего бакалавра по соответствующему направлению подготовки

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов

способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

3.Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе (астр.)		
			СРС	Контактная работа с преподавателем	Всего
7 семестр					
ОПК-8.1, ПК-3.1, ПК-3.2, ПК-10.2	Самостоятельное изучение литературы и источников	Собеседование	15	1	16
ОПК-8.1, ПК-3.1, ПК-3.2, ПК-10.2	Подготовка к лабораторным занятиям	Защита ЛР	13	1	14
Итого за 7 семестр			28	2	30
Итого			28	2	30

4.Порядок выполнения самостоятельной работы студентом

4.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и

понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют ***четыре основные установки в чтении научного текста***:

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

4.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям

Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

4.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

4.4. Методические рекомендации по подготовке к экзаменам и зачетам

Изучение многих общепрофессиональных и специальных дисциплин завершается экзаменом. Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами интервал 3-4 дня. Не следует думать, что 3-4 дня достаточно для успешной подготовки к экзаменам.

В эти 3-4 дня нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом студентов познакомят с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня; сон не менее 8 часов в сутки, занятия заканчиваются не позднее, чем за 2-3 часа до сна. Оптимальное время занятий - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неутомительные занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо во время ее восстановить (переписать ее на кафедре), обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к экзаменам у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных сигналов.

Вначале следует просмотреть весь материал по сдаваемой дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения, используя при этом листы опорных сигналов.

Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

5.Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.

Предусмотрены следующие виды контроля: собеседование, оценка доклада, оценка презентации, оценка участия в круглом столе, оценка выполнения проекта.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

1. Перечень основной литературы:

– Фридман, А.Л. Язык программирования Си++ Электронный ресурс : учебное пособие / А.Л. Фридман. - Язык программирования Си++,2021-01-23. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 218 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0017-5, экземпляров неограничено

– Агафонов, Е. Д. Прикладное программирование / Е.Д. Агафонов ; Г.В. Ващенко. - Красноярск : Сибирский федеральный университет, 2015. - 112 с. - ISBN 978-5-7638-3165-8, экземпляров неограничено

2. Перечень дополнительной литературы:

– Зюзьков, В.М.; Программирование Электронный ресурс : учебное пособие / В.М. Зюзьков. - Томск : Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013. - 186 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-4332-0141-5, экземпляров неограничено

3. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

– Методические указания по выполнению практических работ по дисциплине «Современные языки программирования» (электронный ресурс).

– Методические рекомендации по организации проведению самостоятельной работы по дисциплине «Современные языки программирования» (электронный ресурс).

4. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

– http://el.ncfu.ru/ – система управления обучением ФГАОУ ВО СКФУ. Дистанционная поддержка дисциплины «Цифровая грамотность и обработка данных»

– http://www.intuit.ru – Интернет-Университет Компьютерных технологий.