Федеральное государственное бюджетное образовательное учреждение
высшего образования
Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

**УТВЕРЖДАЮ**
**декан факультета вычислительной**
**математики и кибернетики**

**_____ /И.А. Соколов /**
**«27» сентября 2022г.**

# ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

**по дисциплине**

**Практикум на ЭВМ (Программирование)**

---

**Уровень высшего образования:**

**бакалавриат**

**Направление подготовки / специальность:**

**01.03.02 "Прикладная математика и информатика" (3++)**

**Направленность (профиль) ОПОП:**

**Искусственный интеллект и анализ данных**

**Форма обучения:**

**очная**

Рассмотрен и утвержден

*на заседании Ученого совета факультета ВМК*

(протокол №7, от 27 сентября 2022 года)

Москва 2022

## 1. ФОРМЫ И ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

В процессе и по завершении изучения дисциплины оценивается формирование у студентов следующих компетенций:

| **Планируемые результаты обучения по дисциплине (модулю)** | | |
|---|---|---|
| Содержание и код компетенции. | **Индикатор (показатель) достижения компетенции** | **Планируемые результаты обучения по дисциплине, сопряженные с индикаторами достижения компетенций** |
| ОПК-5. Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения | ОПК-5.1. Разрабатывает программу для решения задачи с использованием языка высокого уровня.<br><br>ОПК-5.2. Умение создавать, тестировать и отлаживать программы на языках программирования высокого уровня на компьютере.<br>ОПК-5.3. Навыки написания качественного и хорошо документированного программного кода | Знать:<br>задачи безусловной и условной численной оптимизации функций многих переменных, задачи комбинаторной оптимизации и прикладные методы их решения, язык компьютерной вёрстки TeX с расширениями LaTeX и Beamer, современные пакеты прикладных программ для решения задач оптимизации, обработки данных и инженерных расчётов на примере языка программирования Python и сопутствующих ему библиотек Pandas, Numpy, Scipy.<br>принципы организации и функционирования компьютеров;<br>особенности архитектуры процессора персонального компьютера (ПК);<br>базовые концепции языка ассемблера ПК;<br>технологию выполнения программ на ЭВМ.<br>основные понятия, алгоритмы и методы организации управления процессами в операционной системе UNIX;<br>основные понятия, алгоритмы и методы организации взаимодействия процессов в операционной системе UNIX;<br>основные понятия, алгоритмы и методы организации работы с файлами;<br>основные понятия языка C++;<br>понятие перегрузки функций и операций;<br>основные понятия наследования, единичное и множественное наследование;<br>понятие механизма виртуальных функций;<br>понятие об обработке исключений в C++;<br>основы параметрического полиморфизма, шаблоны функций и классов;<br>основы теории формальных языков и грамматик;<br>классификацию формальных грамматик и языков по Хомскому;<br>понятие приведенных грамматик, алгоритм получения приведенной КС-грамматики;<br>основы теории трансляции;<br>алгоритм преобразования недетерминированного конечного автомата в детерминированный;<br>метод рекурсивного спуска; |

| | | |
|---|---|---|
| | | понятие синтаксически управляемого перевода; ПОЛИЗ;<br>задачи и схемы работы лексического и синтаксического анализаторов.<br>Уметь:<br>применять на практике средства теории оптимизации для решения задач соответствующих классов;<br>реализовывать алгоритм на языке программирования Python с использованием и без использования сопутствующих ему библиотек;<br>составлять отчёт и презентацию с помощью языка компьютерной вёрстки TeX.<br>находить и анализировать научно-техническую информацию по изучаемому предмету, в том числе в электронном виде;<br>составлять программы на машинном языке для модельных ЭВМ различных архитектур;<br>разрабатывать программы на языке Ассемблере ПК;<br>организовывать связь программы и процедур на Ассемблере в соответствии со стандартными соглашениями о связях;<br>организовывать связь программ, написанных на языке высокого уровня (Free Pascal) и на машинно-зависимом языке (Ассемблер).<br>разрабатывать алгоритмы для решения типовых задач, оценивать сложность полученных алгоритмов,<br>реализовывать программы на языке Си с использованием системных вызовов ОС UNIX,<br>тестировать написанные самостоятельно программы на соответствие исходным требованиям;<br>находить, анализировать и контекстно обрабатывать научно-техническую информацию;<br>создавать алгоритмические модели типовых задач, проводить спецификацию задачи,<br>реализовывать программы на языке C++,<br>оценивать сложность полученных алгоритмов;<br>разрабатывать алгоритмы для решения задач системного и прикладного программирования.<br>применять алгоритм получения приведенной КС-грамматики;<br>применять метод преобразования недетерминированного конечного автомата к детерминированному;<br>применять метод рекурсивного спуска к КС-грамматикам;<br>применять на практике основные методы теории трансляции. |

| | | |
|---|---|---|
| | | Владеть:<br>основными подходами к решению задач теории оптимизации современными программными средствами, навыками использования современных пакетов прикладных программ.<br>навыками анализа и понимания архитектуры конкретной ЭВМ;<br>навыками использования компонент системы программирования для подготовки и запуска на выполнение программы на Ассемблере.<br>основами алгоритмизации, пониманием методов построения алгоритмов на основе разбиения задачи на подзадачи;<br>навыками программирования на языке Си с использованием функций стандартной библиотеки языка Си, а также с использованием библиотеки системных вызовов ОС UNIX;<br>навыками написания программ для работы с текстовыми и бинарными файлами;<br>базовыми навыками разработки и реализации параллельных программ, организации взаимодействия процессов с использованием средств, предоставляемых ОС UNIX;<br>навыками работы с пользовательским интерфейсом ОС UNIX.<br>навыками решения практических задач на языке C++;<br>навыками решения практических задач по теории формальных грамматик. |

**1.1. Текущий контроль успеваемости**

Текущий контроль успеваемости осуществляется путем оценки результатов выполнения заданий практических (семинарских) занятий, самостоятельной работы, предусмотренных учебным планом и посещения занятий/активность на занятиях.

В качестве оценочных средств текущего контроля успеваемости предусмотрены:

Практическая работа
Контрольная работа

**1 семестр**
Контрольная работа

**Вариант 1**

I. Переведите число $123_{10}$ в двоичную и шестнадцатеричную системы счисления. (1б)

II. Переведите число $24{,}125_{10}$ в двоичную и шестнадцатеричную системы счисления. (2б)

III. ***Какую функцию вычисляет машина Тьюринга*** с программой, представленной таблицей, приведенной ниже, если на ленте записано подряд $x + 1$ единиц, а слева и справа от них – символы $e$. Маркер находится против левой единицы. Таблица имеет вид:

| | ***q*1** | ***q*2** | ***q*3** | ***q*4** | ***q*5** | ***q*6** |
|---|---|---|---|---|---|---|
| ***e*** | *e q*1 *S* | *e q*3 *S* | | 1 *q*3 *S* | 1 *q*3 *S* | 1 *q*3 *S* |
| **1** | *e q*2 *R* | *e q*6 *R* | | 1 *q*5 *S* | 1 *q*4 *R* | 1 *q*3 *S* |

***Покажите по шагам****, как вычисляется результат выполнения приведенной выше программы* машины для числа **15**, записав число в предложенной системе. (2б)

IV. Построить машину Тьюринга, прибавляющую 3 к целому неотрицательному числу (вычислить функцию $F(x) = x + 3$). $A = \{0, 1, e\}$ (операции выполняются в двоичной системе). (3б)

V. Каков результат действия на слово $P = nnnnnnnnn$ нормального алгорифма Маркова со схемой (***показать по шагам***, как применяются формулы подстановок).
$Z = | nnn \rightarrow abc \mid ca \rightarrow ac \mid ba \rightarrow nnnn |$ ? (2б)

**Вариант 2**

I. Переведите число $254_{10}$ в двоичную и шестнадцатеричную системы счисления. (1б)

II. Переведите число 0,03125 в двоичную и шестнадцатеричную системы счисления. (2б)

III. ***Какую функцию вычисляет машина Тьюринга*** с программой, представленной таблицей, если на ленте записано подряд $x + 1$ единиц, а слева и справа от них – символы $e$. Маркер находится против левой единицы. Таблица имеет вид:

| | ***q*1** | ***q*2** | ***q*3** | ***q*4** | ***q*5** |
|---|---|---|---|---|---|
| ***e*** | *e q*1 *S* | *e q*4 *S* | 1 *q*4 *R* | 1 *q*5 *R* | |
| **1** | *e q*2 *R* | *e q*3 *R* | 1 *q*3 *R* | | |

***Покажите по шагам****, как вычисляется результат* выполнения программы построенной машины для числа **7**. (2 б)

IV. Построить машину Тьюринга, умножающую на 4 целое положительное число, записанное в двоичной системе (вычислить функцию $F(x) = 4 \times x$). $A = \{0, 1, e\}$ (операции выполняются в двоичной системе). (3 б)

V. Каков результат действия на слово $P = hgdfasdfghjgdsadfgg$ нормального алгорифма Маркова со схемой (***показать по шагам***, как применяются формулы подстановок).
$F = | fg \rightarrow sa \mid df \rightarrow . hg \mid j \rightarrow \Lambda \mid gg \rightarrow . d |$ ? (2б)

**2 семестр**

Типовые контрольные задания или иные материалы для проведения текущего контроля успеваемости.

1. Написать программу для учебной машины УМ-3. Эта программа должна сначала вводить целочисленный массив X длины 100, затем печатать число S, равное количеству одновременно отрицательных и кратных трём элементов массива X. При записи кодов операций использовать мнемонические обозначения.
2. Выписать вид внутреннего машинного представления целой переменной X (в двоичном или шестнадцатеричном виде):

```
X   dw -1023
```

3. Написать полную программу на Ассемблере, которая вводит (по **ininт**) целое знаковое число X в формате **dd** и выводит (по **outword**) число значащих *чётных* цифр (т.е. '0', '2', '4', '6', '8') в десятичной записи значения числа X. Цифра является значащей, если её удаление меняет величину числа.
4. Написать полную программу на Ассемблере, которая вводит текст до точки и выводит (по **outword**) сумму *нечётных цифр*, расположенных в этом тексте после первой "*". Считать, что таких цифр не более MaxLongword.
5. Пусть на Паскале дано описание типа массива:

```
const n=5000; type MAS=array[1..n,1..n] of char;
```

Написать на Ассемблере *процедуру* со стандартными соглашениями о связях, которая получает в качестве параметров адрес массива типа MAS и длину массива N. Процедура должна все символы-*цифры* на главной диагонали этой матрицы заменить на символы "+". Привести пример вызова этой процедуры.
6. Написать макрос с заголовком

```
First_1 macro X
```

параметр которого может быть только форматов m8, m16 или m32. Макрос переставляет в начало все "1" (биты со значением единица)
во внутреннем машинном представлении X, например, для `X = 10101010b` необходимо получить `X = 11110000b`. Макроопределение должно настраиваться на тип параметра..
7. Написать на Ассемблере неголовной модуль, содержащий описание процедуры без параметров с именем Del_3. Эта процедура должна уменьшать в три раза значение знаковой переменной размером в слово (**dw**) с именем Perem3, описанной в каком-то другом модуле.

3 семестр

Пример контрольных работ

**Пример задания № 1. Работа с файлами. Стандартная библиотека**

Программа. Дан файл и строка. Все вхождения строки в файл (в том числе и в качестве подстроки) удалить. Имя файла и строка задаются в командной строке. Строка с пробелами при передаче ее в командной строке должна быть заключена в двойные кавычки.

**Пример задания № 2. Работа с файлами. Системные функции**

Написать программу. В командной строке передается имя файла и число **n**.
а) Вывести на экран **n** первых строк файла, если строк в файле меньше **n**, вывести их все.
б) Если файла с заданным именем еще не существует, создать его и заполнить **n** строками, введенными со стандартного ввода.
Для работы с файлом использовать только системные функции.

**Пример задания № 3. Реализация команд UNIX.**

Сортировка файлов **sort [ -r +n –m -o] файлы**

Сортировать строки каждого файла в лексикографическом порядке.

**sort -r файлы -** сортировка в обратном порядке.

**sort +n файлы –** сортировать файл, начиная с **n**-ой строки.

**sort -m файлы** – слияние исходных (предварительно отсортированных ) файлов.

**sort -o выходной файл** - результат направляется не на стандартный вывод (как происходит по умолчанию), а в **выходной файл**, который может совпадать с одним из исходных.

Порядок опций при задании команды может быть любым, и любая из них (в том числе и все) при запуске программы может отсутствовать.

**Пример задания № 4. Взаимодействие процессов. Каналы**

В командной строке передается имя файла.

Процесс-отец создает сына. Отец считывает со стандартного ввода строку (длина не больше 20 символов) и передает эту строку процессу-сыну. Сын выводит в заданный файл полученную строку. После этого отец выводит в файл строку из 10 '=' и снова переходит к вводу. Работа продолжается до ввода команды «exit». Синхронизировать работу с помощью аппарата каналов.

*Подсказка.* После каждого вывода рекомендуется сбрасывать буфер с помощью вызова функции fflush(). Завершение процессов также должно быть аккуратным – с закрытием файла.

**Пример задания № 5. Взаимодействие процессов. Сигналы**

Написать программу. Работают два процесса: отец и сын.

Отец выводит на экран "Father", а сын - "Son". Вывод строго чередуется и должен происходить многократно (Father Son Father Son Father Son и т.д.) в течение 2 секунд.

Через 2 секунды сын завершает работу, а отец продолжает выводить "Father", но с интервалом в 1 секунду. Завершается отец по Ctrl-C. Не должно остаться процессов –зомби.

**Пример задания № 6. Моделирование элементов работы командного интерпретатора Shell**

**Реализация конвейера.**

Написать программу, которая осуществляет запуск конвейера из произвольного количества процессов. Команда считывается со стандартного ввода. Правильная команда описывается следующими правилами:

<Команда> → <Конвейер>{<перенаправление ввода/вывода>}
<перенаправление ввода/вывода> →
{<перенаправление ввода > } <перенаправление вывода> |
{<перенаправление вывода>}<перенаправление ввода >
<перенаправление ввода > → '**<**' файл
<перенаправление вывода> → '**>**' файл | '**>>**' файл
<Конвейер>→ <Простая команда> {'|' <Конвейер>}
<Простая команда>→ <имя команды><список аргументов>

Примеры команды: **ls –a –l |wc| wc**

**cat | sort -r <file1 >file2**

Пробелы между отдельными элементами команды (аргументы, имена файлов, '|', '>', '>>', '<') допустимы в произвольном количестве.

**pr1 | …| prN** – конвейер: стандартный вывод всех команд, кроме последней, направляется на стандартный ввод следующей команды конвейера. Каждая команда выполняется как самостоятельный процесс (т.е. все pri выполняются параллельно). Управляющий процесс ожидает завершения последней команды. Не должно оставаться процессов ≪зомби≫.

**Пример задания № 7. IPC**

Сервер – 2 клиента.

Сервер в командной строке получает имя файла. Длина строк в файле ограничена MAXLEN.

Сервер читает строки из файла и передает их клиентам через разделяемую память.

Клиент считывает из разделяемой памяти строку, дописывает в начало строки свои идентификационные данные (например, свой PID) и выводит строку в файл-результат (имя определено заранее).

После того, как оба клиента считали строку, сервер помещает в разделяемую память следующую строку файла и т.д. Работа продолжается до конца исходного файла.

Вывод в файл-результат осуществляется клиентами строго поочередно.

Синхронизацию доступа к файлу реализовать с помощью аппарата семафоров.

Разделяемая память и семафор(ы) должны быть корректно удалены по окончании работы.

4 семестр

**Контрольная работа № 1 «Язык Си++».**

Во всех задачах, где это требуется, предполагается наличие необходимых включений и директивы using namespace std.

1. Что напечатает следующая программа?

```
class I {
        int i;
        public: I() : i(6) { cout << "owl" << endl; }
        I(int a) : i(a) { cout << "sheep " << i << endl; }
        I(const I & other) : i(other.i) { cout << "horse " << i << endl; }
        ~I() { cout << "wolf" << endl; }
        int Get() { return i; }
        void operator*=(const I & op) { i*=op.i; }
    };

void f (I x, I & y) {
                    x *= 1; y *= x;
                }

int main() {
            I i1;
            I i2(3);
            i1 *= 7;
            f(i1, i2);
            cout << i1.Get() << ' ' << i2.Get()<< endl;
            return 0;
      }
```

2. Объект x принадлежит классу X, для которого определены конструктор преобразования типа int в класс X и функция преобразования класса X в тип int. .Описать прототипы двух перегруженных функций h из некоторой области видимости, для которых будут верны следующие обращения к ним:

```
h ("333");
h (x, 0);
h (1, "dd );
h (0,0);
h(x, "nn");
```

3. Найдите ошибки в программе и объясните, в чем они заключаются.

```
class table{
            int size;
```

```
        int priority;
        public:  table(int s=0,int p):size(s),priority(p){ }
                 virtual void print()=0;
       };

class stud_table:  public table{
                    char * name;
                    int gr;
                    public: void print(){cout<<"students table"<<endl;}
                    ~stud_table(){delete []name;}
      };

class asp_table: protected table{
                                char* thesis;
            };

int main(){
          table t;
          stud_table st;
          table * tp = &st;
          tp = new asp_table();
          stud_table * stp = &st;
          cout<<"Program"<<endl;
          return 0;
      }
```

4. Что будет выдано в стандартный канал вывода при работе следующей программы?

```
struct X;
void f (X & x, int n);
int const P = 1;
int const Q = 1;
int const R = 1;

struct X {
      X() { try { f(*this, -1);
                  cout << 1 << endl;
                  }
                 catch (X) { cout << 2 << endl; }
                 catch (int) { cout << 3 << endl; }
                }
       X (X &) { cout << 4 << endl; }
      ~X () { cout << 5 << endl; }
      };

struct Y: X {
            Y () { f(*this, 1);
                   cout << 6 << endl;
                   }
            Y (Y &) { cout << 7 << endl; }
            ~Y () { cout << 8 << endl; }
             };
void f (X & x, int n) {
                    try { if (n < 0) throw x;
```

```
                    if (n > 0) throw 1;
                    cout << 9 << endl;
                     }
                catch (int) { cout << 10 << endl; }
                catch (X& a) { cout << 11 << endl;
                               f(a, 0);
                               cout << 12 << endl;
                               throw;
                              }
            }

int main() {
            try { Y a; }
            catch (...) { cout << 13 << endl;
                          return 0;
                        }
            cout << 14 << endl;
            return 0;
          }
```

5. Для приведённой ниже программы описать функцию f (), которая, получая в качестве параметра ссылку на объект базового класса A, возвращает ссылку на объект производного класса C, полученную наиболее безопасным образом, а в случае невозможности приведения типов корректно завершает программу.

```
struct A {
           virtual void z () {}
         };

struct B: A { };

struct C: B {
            int x;
            C (int n = 3) { x = n; }
 };

C & f (A & ra);

int main () {
            C c, & pc = f (& c);
            cout << pc.x << endl;
            return 0;
            }
```

6. Для класса рациональных дробей с числителями и знаменателями некоторого интегрального типа template <class T> class fr { T n; T d; ... }; описать два варианта (методом класса и функцией-другом этого класса) реализации вне класса операций '*', выполняющей умножение одной рациональной дроби на другую.

**Контрольная работа № 2 «Формальные грамматики и языки. Элементы теории трансляции».**

1. а) Построить приведенную грамматику, эквивалентную заданной грамматике G:

S → aAb | CB
B→ bB
A→ aA | D
C→ cCc | B | c
D→ d

б) Каков тип получившейся грамматики?
в) Какой язык порождает грамматика?
г) Каков тип этого языка?

Для типов грамматики и языка указать максимально возможный номер по Хомскому.

2. Дана автоматная леволинейная грамматика G:

S → C⊥
A→ Aa | a
B→ Aa | b
C→ Ba
а) Детерминирован ли разбор по этой грамматике? Обосновать ответ.
б) Построить по грамматике конечный автомат и преобразовать его к детерминированному виду по алгоритму НКА→ДКА
в) Построить по получившемуся ДКА праволинейную автоматную грамматику.
г) Эквивалентны ли исходная и получившаяся автоматные грамматики? Обосновать ответ.

3. Дана КС-грамматика G:

S → A | B
A→ aA | ε
B→ bB | cC | ε
C→ aABC | c

а) Преобразовать грамматику G к неукорачивающей КС-грамматике с помощью алгоритма устранения пустых правых частей в КС грамматике (КС→НКС).
б) Применим ли метод рекурсивного спуска к исходной грамматике? Ответ обосновать.

4. Построить грамматику выражений, содержащих:
- целочисленные бинарные операции + и –
- односимвольные целые числа 1 и 3
Добавить в грамматику действия (только вида cout << "символ"; ) по переводу заданных выражений в ПОЛИЗ во время анализа РС-методом. Приоритет операций + и – должен быть задан построенной грамматикой таким образом, чтобы ПОЛИЗ выражения:
3 -- 1 + 1
совпадал с ПОЛИЗом выражения со скобками:
(3 – (1 + 1))

### 1.2. Промежуточная аттестация

Промежуточная аттестация осуществляется в форме зачета
В качестве средств, используемых на промежуточной аттестации предусматривается:
Билеты

### 1.3. Типовые задания для проведения промежуточной аттестации

1 семестр

1) Понятие системы счисления. Связь между системами счисления и алгоритмы перевода.
2) Понятие типа данных. Представление данных в памяти компьютера. Форматы представления числовых данных.
3) Конструирование типов. Рекурсивные типы данных: определение, примеры.
4) Понятие алгоритма и свойства алгоритмов. Примеры.
5) Способы записи алгоритмов. Примеры.
6) Методы разработки алгоритмов. Суперпозиция, итерация и рекурсия. Примеры.
7) Формализация понятия алгоритма: машина Тьюринга. Представление машин Тьюринга с помощью диаграмм. Табличное представление программ машины Тьюринга. Сравнение. Примеры. Композиция машин Тьюринга. Связь с методами разработки алгоритмов.
8) Формализация понятия алгоритма: нормальные алгорифмы Маркова, определение и выполнение. Примеры.
9) Проблема алгоритмической разрешимости. Примеры алгоритмически неразрешимых задач.
1. Понятие сложности задачи и классы сложности задач. Типы задач. Понятие сводимости, полиномиальная сводимость.

2 семестр

1. Системы счисления.
2. Машины Тьюринга.
3. Нормальные алгоритмы Маркова.
4. Металингвистические формулы.
5. Синтаксические диаграммы.
6. Регулярные типы.
7. Комбинированные типы данных.
8. Процедуры и функции.
9. Ссылки, списки.
10. Стеки, очереди.
11. Деревья.
12. Двоичные деревья поиска.
13. АВЛ-деревья.

3 семестр

Примеры контрольных заданий
1, Дан однонаправленный список, элемент данных типа **int.**

Написать функцию (от 2 параметров) для удаления из списка всех элементов, значения которых равны заданному числу.

2. Написать функцию, параметр – имя файла. Функция должна в данном файле поменять местами первую и последнюю строки. Использовать функции стандартной библиотеки. Длина строк в файле неограниченна.

3. В командной строке передаются имена нескольких исполняемых файлов. Если их меньше двух, ничего не делать. Иначе запустить эти файлы сначала на параллельное исполнение. Когда они все выполнятся, запустить еще раз последние 2 из них, но уже последовательно. Не должно остаться процессов «зомби».

4. Написать программу. Работают 2 процесса. Отец читает со стандартного ввода по строке (длина строки не больше MAXLEN), передает строку сыну. Сын выводит на экран длину строки. Только **после вывода** длины отец **считывает** очередную строку. Работа заканчивается по второму нажатию ^C. **Синхронизировать работу с помощью сигналов**. Не должно остаться процессов «зомби».

5. Написать программу. Работают 2 процесса. Отец читает со стандартного ввода имя исполняемого файла, передает его сыну. Сын запускает этот файл на исполнение. Только после того, как исполняемый файл завершит работу, сын сообщает об этом отцу, и отец считывает следующее имя. Работа заканчивается по второму нажатию ^C. Синхронизировать работу с помощью сигналов. Не должно остаться процессов «зомби».

4 семестр

**Вопросы к зачету**

1. Определение типа по Хомскому заданной грамматики.
2. Определение типа языка, порождаемого заданной грамматикой.
3. Построение грамматики определенного типа, порождающей заданный язык.
4. Определение языка, порождаемого заданной грамматикой.
5. Построение приведенной грамматики.
6. Устранение пустых правых частей заданной КС-грамматики.
7. Построение ДС конечного автомата по заданной леволинейной грамматике.
8. Построение анализатора на C++ по заданной ДС.
9. Восстановление леволинейной регулярной грамматики, порождающей язык, распознаваемый конечным автоматом, заданным в виде ДС.
10. Восстановление леволинейной регулярной грамматики по заданному тексту анализатора на C++.
11. Построение ДС по праволинейной грамматике и построение праволинейной грамматики по заданной ДС.
12. Преобразование праволинейной грамматики в эквивалентную леволинейную с помощью ДС.
13. Преобразование НКА в эквивалентный ДКА с помощью соответствующего алгоритма.
14. Определение и обоснование применимости метода рекурсивного спуска к заданной КС-грамматике.
15. Построение (на C++) анализатора методом рекурсивного спуска для заданной КС-грамматики.
16. Восстановление КС-грамматики по заданному анализатору, построенному методом рекурсивного спуска.
17. Определение языка, порождаемого грамматикой с действиями.
18. Дополнение заданной КС-грамматики действиями, позволяющими учесть дополнительные ограничения на цепочки определяемого ею языка.
19. Вставка в заданную (или построенную) грамматику языка $L1$ действий по переводу цепочек этого языка в цепочки языка $L1$ по заданному закону соответствия между цепочками (т. е. реализовать
20. формальный перевод).
21. Определение языков $L1$ и $L1$ по заданной грамматике с действиями, реализующей формальный перевод языка $L1$ в язык $L1$.
22. Запись в ПОЛИЗе заданного фрагмента программы на заданном языке.
23. Восстановление текста на заданном языке заданного фрагмента программы в ПОЛИЗе.

24. Определение, является ли данная запись ПОЛИЗом заданной конструкции.
25. Вставка в грамматику, порождающую некоторую конструкцию языка программирования, действий, осуществляющих синтаксически управляемый перевод этой конструкции в ПОЛИЗ при анализе РС-методом.

**Типовые задачи**

1. Для автоматизированной системы регистрации на курсы кройки и шитья описать классы Course и Student. Класс Course описывает один конкретный курс. В нем должно быть предусмотрено хранение информации об общем количестве слушателей этого курса, максимальное допустимом количестве слушателей для этого курса. У слушателя курсов (Student) должны быть определены фамилия, имя, номер паспорта, информация о курсе, на котором он учится. Предусмотреть функции добавления слушателя и исключения его с курсов. Один и тот же слушатель не может учиться на нескольких курсах. Написать функцию, не являющуюся членом указанных классов, которая для двух параметров Student проверяет, учатся ли они на одном курсе.

2. Описать шаблонную функцию max, которая возвращает значение максимального элемента для заданного массива целых чисел (int), длинных целых чисел (long), массива вещественных чисел (double) или массива строк (constchar*). При работе со строками, максимальной считать ту строку, которая имеет наибольшую длину.

3. Определить иерархию классов для описания сессии в университете. Базовый класс – event, содержит информацию о дате события, фамилию действующего лица и чисто виртуальную функцию print_res, печатающую информацию о событии; классы-наследники – test (зачет), exam(экзамен). В них должны быть определены дополнительные характеристики событий и метод print_res. В функции main() определить сессию как массив указателей на события, проинициализировать элементы массива и распечатать информацию об экзаменах и зачетах.

4. Написать лексический анализатор на языке C++ по заданной автоматной грамматике:

Gleft= ⟨ {a,b,⊥}, {S,A,B,C}, P, S⟩ с правилами P:

S→ C⊥
C→ Ab | Ba
A→ a | Ca
B→ b | Cb

5. Написать синтаксический анализатор на языке C++ методом рекурсивного спуска по заданной грамматике G:

S→ ABd
A→ a | cA
B→ bA

**Пример билета**

1. Построение (на C++) анализатора методом рекурсивного спуска для заданной КС-грамматики.
2. Написать лексический анализатор на языке C++ по заданной автоматной грамматике:
   Gleft= ⟨ {a,b,⊥}, {S,A,B,C}, P, S⟩ с правилами P:
   S→ C⊥
   C→ Ab | Ba
   A→ a | Ca
   B→ b | Cb

## 2. КРИТЕРИИ ОЦЕНКИ ПО ДИСЦИПЛИНЕ

| ШКАЛА И КРИТЕРИИ ОЦЕНИВАНИЯ результатов обучения (РО) по дисциплине | | | | |
|---|---|---|---|---|
| Оценка<br><br>виды оценочных средств | **2 (не зачтено)** | **3 (зачтено)** | **4 (зачтено)** | **5 (зачтено)** |
| **Знания**<br>**(виды оценочных средств: приведены в п. 1.2.)** | Отсутствие знаний | Фрагментарные знания | Общие, но не структурированные знания | Сформированные систематические знания |
| **Умения**<br>**(виды оценочных средств: приведены в п. 1.2.)** | Отсутствие умений | В целом успешное, но не систематическое умение | В целом успешное, но содержащее отдельные пробелы умение (допускает неточности непринципиального характера) | Успешное и систематическое умение |
| **Навыки**<br>**(владения, опыт деятельности)**<br>**(виды оценочных средств: приведены в п. 1.2..)** | Отсутствие навыков (владений, опыта) | Наличие отдельных навыков (наличие фрагментарного опыта) | В целом, сформированные навыки (владения), но используемые не в активной форме | Сформированные навыки (владения), применяемые при решении задач |