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

Алгоритмы и алгоритмические языки

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-2. Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач	ОПК-2.1. Знание приемов написания и анализа алгоритмов и компьютерных программ; ОПК-2.2. Способность анализировать и конструировать конкретные алгоритмы на языке высокого уровня для решения разнообразных математических задач на компьютере. ОПК-2.3. Знание парадигм структурного, процедурно-модульного и объектно-ориентированного программирования на языке высокого уровня.	Знать: 1. неформальное и формальные определения понятия «алгоритм» 2. основные способы конструирования алгоритмов 3. определения эквивалентности машин Тьюринга 4. существование универсальной машины Тьюринга 5. существование алгоритмически неразрешимых проблем 6. методы доказательства алгоритмической неразрешимости 7. язык программирования Си, его системные библиотеки, структуру Си-программы. 8. базовые алгоритмы решения задач сортировки, поиска, топологической сортировки, работы с текстами. 9. основные структуры данных: стек, очередь, список, дерево и т.п. Уметь: 1. строить алгоритмы для решения простых задач в алгоритмических системах Тьюринга и Маркова 2. строить универсальную машину Тьюринга 3. доказывать алгоритмическую неразрешимость конкретных проблем

		4. составлять и отлаживать программы на языке Си 5. использовать системные библиотеки языка Си 6. применять базовые алгоритмы и основные структуры данных, изучаемые в курсе, при разработке программ. 7. оценивать сложность алгоритмов при их выборе. Владеть: 1. современной технологией разработки и отладки программ на языке Си.

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:

коллоквиум

Пример заданий коллоквиума

1. Пусть определены следующие переменные:
int x = 2, y = 4, z = 6;
short n = 0x7FFF; **unsigned short** m = 0x7FFFU;
struct S { **struct** P { **int** x, y; } p; **struct** S *n; } s = { { -10, -20 },
&s };
unsigned int a[10] = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 };
unsigned int *p, **pp;
Для каждого из **8** отдельных независимых выражений указать ***его значение*** и ***побочные эффекты*** (если они есть) либо “ошибка”, если выражение ошибочно.
1) x | y ^ z
2) n + 4
3) m <<= 1
4) s.p.x = x + s.n->p.y
5) *(a + 4 * x)
6) p = a + 1, **(pp = &p)
7) *pp = &a[2], *(*pp + y + a[2])
8) p = a + a[3], s.p.y = ++p - a

УКАЗАНИЕ.
В задачах 2-4 определите все необходимые переменные и типы.
Считайте, что вызовы функций выделения памяти всегда заканчиваются успешно.

2. Напишите функцию, принимающую единственный параметр – массив строк (указателей типа **char ***). Последний элемент массива содержит нулевой указатель. Функция должна возвратить

указатель на размещенный в динамической памяти массив структур, в которых первое поле есть указатель на копию одной из входных строк, а второе поле – количество раз, которое данная строка встречалась во входном массиве (с учетом регистра символов). Одинаковые строки (с учетом регистра символов) должны содержаться в выходном массиве только в одном экземпляре, поле с указателем может указывать на любую из них. Выходной массив должен быть отсортирован сначала по убыванию по второму полю, а затем в лексикографическом порядке по первому полю.

3. Напишите функцию, которая удаляет из односвязного списка все элементы с нечетными номерами. Порядок следования всех остальных элементов должен остаться неизменным. Память, занятая удаляемыми элементами списка, должна быть освобождена с помощью вызова функции free. Считайте, что заголовочного элемента в списке нет и элементы списка нумеруются с 1. Ваша функция должна соответствовать прототипу
void remove_odd (struct list **);
где struct list описывает элемент списка с целочисленным ключом.

4. Напишите функцию, принимающую три параметра – массив целых чисел, его длину и указатель на целое число. Элементы массива образуют (возможно пустое) двоичное дерево (для элемента a[i] его детьми в двоичном дереве являются элементы a[2*i+1] и a[2*i+2], элемент a[0] – корень дерева). Функция должна возвратить указатель на построенное в динамической памяти (в виде рекурсивной ссылочной структуры) двоичное дерево, в котором для каждого узла детьми являются те же числа, что и во входном массиве (или нулевой указатель для пустого дерева). По указателю, который является третьим параметром, необходимо записать 1, если построенное дерево является двоичным деревом поиска, и 0 в противном случае.

1.2. Промежуточная аттестация

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

1.3. Типовые задания для проведения промежуточной аттестации

Вопросы к экзамену

1. Алгоритмы. Формализация понятия алгоритма. Машина Тьюринга (МТ).Нормальные МТ. Диаграммы Тьюринга: диаграммы элементарных МТ, правила композиции диаграмм, примеры диаграмм, построение таблицы МТ по диаграмме.
2. Универсальная МТ. Построение универсальной МТ. Проблемы останова и самоприменимости. Нормальные алгоритмы Маркова.
3. Тезис Тьюринга-Черча. Стандарты и роль стандартизации. Первая программа на Си. Структура программного файла. Системные библиотеки и их использование.
4. Типы данных языка Си: целые, логические, символьные, с плавающей точкой. Представление в памяти переменных целочисленных типов. Переменные: класс памяти, область действия.
5. Арифметические и логические выражения. Ленивое вычисление логических выражений.
6. Точки следования. Форматный ввод-вывод. Приведение типов при вычислении выражений (явное и неявное).
7. Символьный тип и обработка символьных данных.
8. Инициализация массивов. Строки. Обработка строк. Операция sizeof
9. Адресная арифметика. Массивы и указатели. Преобразования типа указателей.
10. Определение и объявление функции. Организация автоматической памяти. Передача параметров. Рекурсия. Хвостовая рекурсия.
11. Указатели на функцию. Побитовая обработка данных. Структуры. Указатели на структуры. Составные инициализаторы структур.
12. Анонимные объединения и структуры. Битовые поля. Перечисления. Схема компиляции программ на языке Си. Препроцессор. Директивы препроцессора. Компоновка. Классы памяти и компоновки. Динамическое выделение и освобождение памяти.
13. Массив переменного размера в составе структуры. Отладка программ. Инструменты поиска ошибок с динамической памятью. Представление данных с плавающей точкой. Стандарт IEEE 754.
14. Потеря точности при сложении и вычитании. Выбор правильной последовательности вычислений. Опции компилятора gcc для вычислений с плавающей точкой. Понятие о сложности алгоритмов. Поиск подстроки по образцу. Простейший алгоритм.
15. Префикс-функция и ее вычисление. Сложность вычисления префикс-функции и алгоритма КМП. Организация типа данных «стек» на динамической памяти.
16. Использование стека для построения обратной польской записи. Очередь. Списки. Добавление нового элемента в начало и в конец списка. Поиск элемента в списке. Удаление заданного элемента из списка: через возврат указателя на новый список, через передачу двойного указателя.
17. Алгоритм топологической сортировки Вирта. Простейшие алгоритмы сортировки (выбором, вставками, обменами). Оценка сложности алгоритмов сортировки.
18. Двоичные деревья и его обходы (сначала в глубину, сначала в ширину). Нерекурсивный симметричный обход двоичного дерева. «Прошитое» двоичное дерево и его обход.

19. (поиск элемента, минимальный и максимальный элементы, следующий элемент, вставка и удаление элемента).
20. Оценка высоты дерева Фибоначчи. АВЛ-деревья. Базовые операции над АВЛ-деревьями. Балансировка АВЛ-дерева. Вставка и удаление элемента в/из АВЛ-дерева. Оценка высоты АВЛ-дерева по дереву Фибоначчи.
21. Самоперестраивающиеся деревья (splay trees). Операция splay (перестраивание). Реализация словарных операций через операцию splay. Реализация операции splay. Сложность словарных операций в splay-деревьях.
22. Обобщение сбалансированных деревьев поиска: ранговые деревья, понятие ранга и ранговой разницы. Ранговые правила для АВЛ-деревьев и красно-черных деревьев.
23. Сложность пирамидальной сортировки. Хеш-таблицы. Хеширование. Хеширование цепочками. Хеширование с открытой адресацией. Сложность словарных операций для хеш-таблиц. Методы построения хеш-функций: деление с остатком, умножение.
24. Линейная/квадратичная последовательность проб, двойное хеширование. Цифровой поиск. Задача цифрового поиска. Деревья цифрового поиска. Вставка в дерево цифрового поиска.

Вариант письменного экзамена по курсу

1. Составьте диаграмму машины Тьюринга (МТ), которая в непустом входном слове над алфавитом A_3={*0, 1, 2*} меняет местами символы в паре подряд стоящих символов (например, входное слово 02121 преобразуется в слово 20211). Пары символов отсчитываются от левого края (первого символа) слова. Лента МТ бесконечна только справа. В начальном состоянии головка МТ обозревает пустую ячейку сразу после входного слова, в конечном состоянии – пустую ячейку сразу после выходного слова. Можно предполагать, что слева от входного слова на ленте есть одна пустая ячейка. При построении можно использовать элементарные машины *l, r, L, R, символ* (сдвиг головки на одну ячейку влево/вправо, сдвиг головки на одно слово влево/вправо, запись символа в ячейку, соответственно).

2. Дано описание:
struct list {**int** key; **struct** list *next;};
Напишите функцию
void move_elts_keys (**struct** list **src, **struct** list **dst),
которая переносит из односвязного списка src в односвязный список dst все элементы с чётными ключами, удаляя их при этом из исходного списка. Перенесенные элементы должны добавляться в начало списка dst в произвольном порядке. Относительный порядок изначальных элементов обоих списков должен остаться неизменным. Считайте, что заголовочного элемента в обоих списках нет, списки могут быть пустыми.

3. Пусть **int** x = 42; **int** y = -105;
Для каждого из **6** отдельных независимых выражений указать ***его значение*** и ***побочные эффекты*** (если они есть) либо “ошибка”, если выражение ошибочно.
1) y %= x / 5
2) x ^ ~y & 65
3) y <<= (--x || --y)
4) (y = 1) && (y = 2)
5) x++ + --y
6) x += y, y = x - y, x -= y

4. Что будет напечатано при выполнении программы:

```
#include <stdio.h>
int x = 2, y = 1;
int a[3] = { 1, 2 };
int f (int x, int **p) {
  static int c = 2;
```

```
 if (c--)
   **p += x;
 else
   ++x;
 return (*p)++ - a - x;
}

int main (void) {
 int x = y, c = 0;  int *p;
 p = &a[x];
 for (int y = 2; y >= c; --y, ++x)
   c = f (y, &p);
 printf ("%d %d %d %d %d\n", a[0], a[1], a[2], x, y);
 return 0;
}
```

5. Перепишите приведенный фрагмент программы с использованием единственного оператора **switch** (дополнительные условные операторы использовать запрещено):

```
if (i == -1)
  j = i;
else if (3 >= i && i >= 1)
  j = 10 - i;
else
  abort ();
```

6. С использованием единственного цикла **for** напишите фрагмент программы, который для массива **int** a[N] формирует в переменной pairs количество пар соседних элементов, имеющих разную четность. Другие операторы циклов и операторы перехода использовать запрещено.

7. Напишите функцию
int scalar (**void**),
которая считывает со стандартного потока ввода непустую последовательность целых чисел, заканчивающуюся нулем (нуль не входит в последовательность, количество ненулевых элементов четно, элементы нумеруются с 1). Функция возвращает значение скалярного произведения двух векторов, составленных из элементов последовательности: первый вектор – из элементов с нечетными номерами, второй вектор – из элементов с четными номерами. В решении можно использовать не более одного оператора цикла. Запрещается использовать операторы перехода.

8. Напишите функцию
void erase (**char** *s, **const char** *w),
на вход которой дается строка ***s*** и слово ***w***. Слово – непустая строка из латинских букв. Строка ***s*** представляет собой последовательность слов, разделенных одним пробелом. Перед первым словом и после последнего пробелов нет. Функция должна удалить из строки ***s*** все слова, в которые слово ***w*** входит как подстрока. Относительный порядок всех остальных слов должен остаться неизменным. Измененная таким образом строка ***s*** должна удовлетворять всем тем же ограничениям, что и входная. Память под все временные массивы должна выделяться динамически и освобождаться в конце работы функции.

9. Дано описание:
struct avltree {**int** x; **struct** avltree *left, *right;};
Напишите функцию
struct avltree *build (**int** h),
которая строит АВЛ-дерево заданной высоты ***h***, содержащее минимальное количество узлов ***n*** из всех АВЛ-деревьев с этой высотой. Значения ключей в элементах АВЛ-дерева должны быть от 1 до ***n***.

10. Нарисуйте дерево цифрового поиска для алфавита **{*В, М, И, К*},** которое содержит следующие ключи: ***КИМ, МИКИ, ВИМ, ИМ, ИК, ВК, МИМ, ВМК, В, МИ, ВМИК, МИМИ.***

2. КРИТЕРИИ ОЦЕНКИ ПО ДИСЦИПЛИНЕ

ШКАЛА И КРИТЕРИИ ОЦЕНИВАНИЯ результатов обучения (РО) по дисциплине				
Оценка виды оценочных средств	**2 (не зачтено)**	**3 (зачтено)**	**4 (зачтено)**	**5 (зачтено)**
Знания **(виды оценочных средств: приведены в п. 1.2.)**	Отсутствие знаний	Фрагментарные знания	Общие, но не структурированные знания	Сформированные систематические знания
Умения **(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 **(владения, опыт деятельности)** **(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