Государственное образовательное учреждение
«Приднестровский государственный университет им. Т.Г. Шевченко»

Рыбницкий филиал
Кафедра «Информатики и программной инженерии»

««УТВЕРЖДАЮ»»
Заведующий кафедрой информатики и
программной инженерии
доцент _____ Л.А. Тягульская
«21» 09 2023г.

Фонд оценочных средств

ПО ДИСЦИПЛИНЕ «ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Направление подготовки

2.09.03.04 Программная инженерия
(Код и наименование направления подготовки)

Профиль подготовки

«Разработка программно-информационных систем»
(наименование профиля подготовки)

Квалификация (степень)

Бакалавр

Форма обучения

очная

Год набора **2022**

Разработал: преподаватель
_____ О.М. Нагаевский
«21» 09 2023г.

Рыбница 2023г.

Паспорт фонда оценочных средств по учебной дисциплине

1. В результате изучения дисциплины «Типы и структуры данных» у обучающихся должны быть сформированы следующие компетенции:

Категория (группа) компетенций	Код и наименование	Код и наименование индикатора достижения универсальной компетенции
Универсальные компетенции и индикаторы их достижения		
Системное и критическое мышление	УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	ИД УК-1.1. Выбирает источники информации, адекватные поставленным задачам и соответствующие научному мировоззрению ИД УК-1.2. Демонстрирует умение рассматривать различные точки зрения на поставленную задачу в рамках научного мировоззрения и определять рациональные идеи ИД УК-1.3. Выявляет степень доказательности различных точек зрения на поставленную задачу в рамках научного мировоззрения
Общепрофессиональные компетенции выпускников и индикаторы их достижения		
	ОПК-1. Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности	ИД ОПК-1.1. Знает основы математики, физики, вычислительной техники и программирования. ИД ОПК-1.2. Умеет решать стандартные профессиональные задачи с применением естественнонаучных и общеинженерных знаний, методов математического анализа и моделирования. ИД ОПК-1.3. Имеет навыки теоретического и экспериментального исследования объектов профессиональной деятельности.
	ОПК-2. Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности	ИД ОПК-2.1. Знает современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности. ИД ОПК-2.2. Умеет выбирать современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности. ИД ОПК-2.3. Имеет навыки применения современных информационных технологий и программных средств, в том числе отечественного производства, при решении задач профессиональной деятельности.
	ОПК-3. Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	ИД ОПК-3.1. Знает принципы, методы и средства решения стандартных задач профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности. ИД ОПК-3.2. Умеет решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом

		основных требований информационной безопасности. ИД опк-3.3. Имеет навыки подготовки обзоров, аннотаций, составления рефератов, научных докладов, публикаций, и библиографии по научно исследовательской работе с учетом требований информационной безопасности.
	ОПК-6. Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов	ИД опк-6.1. Знает основные языки программирования и работы с базами данных, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий. ИД опк-6.2. Умеет применять языки программирования и работы с базами данных, современные программные среды разработки информационных систем и технологий для автоматизации бизнес-процессов, решения прикладных задач различных классов, ведения баз данных и информационных хранилищ. ИД опк-6.3. Имеет навыки программирования, отладки и тестирования прототипов программно-технических комплексов задач.
	ОПК-7. Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой	ИД опк-7.1. Знает основные языки программирования и работы с базами данных, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий. ИД опк-7.2. Умеет применять языки программирования и работы с базами данных, современные программные среды разработки информационных систем и технологий для автоматизации бизнес-процессов, решения прикладных задач различных классов, ведения баз данных и информационных хранилищ. ИД опк-7.3. Имеет навыки программирования, отладки и тестирования прототипов программно-технических комплексов задач.
Обязательные профессиональные компетенции выпускников и индикаторы их достижения		
	ПК-5. Владение навыками использования различных технологий разработки программного обеспечения	ИД пк-5.1. Знает современные технологии разработки ПО (структурное, объектно-ориентированное) ИД пк-5.2. Умеет использовать современные технологии разработки ПО ИД пк-5.3. Имеет навыки использования современных технологий разработки ПО

2. Программа оценивания контролируемой компетенции:

№ п\п	Контролируемые модули, разделы (темы) дисциплины и	Код контролируемой компетенции (или ее части)	Наименование оценочного средства

	их наименование		
1.	**Динамические структуры данных**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Лабораторные работы. Тестирование.
2.	**Абстрактные типы данных (АТД)**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Лабораторные работы. Тестирование.
3.	**Жадные алгоритмы**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Лабораторные работы. Тестирование.
4.	**Графы**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Лабораторные работы. Тестирование.
5	**Алгоритмы сортировок**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Лабораторные работы. Тестирование.
6	**Алгоритмы поиска**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Лабораторные работы. Тестирование.
7	**Промежуточная аттестация**	УК-1, ОПК-1, ОПК-2, ОПК-3, ОПК-6, ОПК-7, ПК-5	Зачет с оценкой.

«УТВЕРЖДАЮ»
зав. кафедрой ИиПИ,
доцент_____ Л. А. Тягульская
«21» 09 2023 г.

Итоговый тест по дисциплине
«Типы и структуры данных»
для студентов II курса
направления «Программная инженерия» профиля подготовки
«Разработка программно-информационных систем»

ДЕ-1.

1. Структура данных представляет собой
a) набор правил и ограничений, определяющих связи между отдельными элементами и группами данных
b) набор правил и ограничений, определяющих связи между отдельными элементами данных
c) набор правил и ограничений, определяющих связи между отдельными группами данных
d) некоторую иерархию данных

2. В чём отличительная особенность динамических объектов?
a) порождаются непосредственно перед выполнением программы;
b) возникают уже в процессе выполнения программы;
c) задаются в процессе выполнения программы

3. Укажите зарезервированное ключевое слово для динамического выделения памяти!
a) create
b) value
c) new
d) malloc

ДЕ-2.

1. Линейный список, в котором доступен только последний элемент, называется
a) стеком
b) очередью
c) деком
d) массивом
e) кольцом

2. Структура данных работа с элементами которой организована по принципу FIFO (первый пришел - первый ушел) это –
a) Стек
b) Дек
c) Очередь
d) Список

3. Линейный последовательный список, в котором включение исключение элементов возможно с обоих концов, называется
a) стеком
b) очередью
c) деком
d) кольцевой очередью

4. В чём особенности очереди?

a) открыта с обеих сторон;
b) открыта с одной стороны на вставку и удаление;
c) доступен любой элемент.

5. В чём особенности стека?
a) открыт с обеих сторон на вставку и удаление;
b) доступен любой элемент;
c) открыт с одной стороны на вставку и удаление.

6. Какую дисциплину обслуживания принято называть FIFO?
a) стек;
b) очередь;
c) дек.

7. Какая операция читает верхний элемент стека без удаления?
a) pop;
b) push;
c) stackpop.

8. Каково правило выборки элемента из стека?
a) первый элемент;
b) последний элемент;
c) любой элемент

9. Как освободить память от удаленного из списка элемента?
a) p=getnode();
b) p=NULL;
c) delete p;
d) p=lst.

10. Как создать новый элемент списка с информационным полем D?
a) p=getnode();
b) p=getnode();p->info=D;
c) p=getnode(); D=lst.

11. Сколько указателей используется в односвязных списках?
a) 1
b) 2;
c) сколько угодно.

12. При удалении элемента из кольцевого списка…
a) список разрывается;
b) в списке образуется дыра;
c) список становится короче на один элемент.

13. Для чего используется указатель в кольцевых списках?
a) для ссылки на следующий элемент;
b) для запоминания номера сегмента расположения элемента;
c) для ссылки на предыдущий элемент ;
d) для расположения элемента в списке памяти.

14. Чем отличается кольцевой список от линейного?
a) в кольцевом списке последний элемент является одновременно и первым;
b) в кольцевом списке указатель последнего элемента пустой;
c) в кольцевых списках последнего элемента нет ;
d) в кольцевом списке указатель последнего элемента не пустой.

15. Сколько указателей используется в односвязном кольцевом списке?
a) 1;
b) 2;
c) сколько угодно.

16. .В каких направлениях можно перемещаться в кольцевом двунаправленном списке?
a) в обоих;
b) влево;
c) вправо.

17. С помощью какой структуры данных наиболее рационально реализовать очередь?
a) стек;
b) список;
c) дек.

18. В памяти ЭВМ бинарное дерево удобно представлять в виде:
a) связанных линейных списков;
b) массивов;
c) связанных нелинейных списков.

19. Элемент t, на который нет ссылок:
a) корнем ;
b) промежуточным;
c) терминальным (лист).

20. Дерево называется полным бинарным, если степень исходов вершин равна:
a) 2 или 0 ;
b) 2;
c) M или 0;
d) M.

21. Укажите, сколько вершин — непосредственных потомков может иметь вершина бинарного дерева.
a) Две
b) Не более двух
c) Не менее одной
d) Любое количество

22. Укажите, сколько вершин — непосредственных потомков может иметь вершина произвольного дерева.
a) Две
b) Не более двух
c) Не менее одной
d) Любое количество

23. Укажите вариант перебора вершин бинарного дерева, называемый инфиксным.
a) Вершина – левое поддерево – правое поддерево
b) Левое поддерево – вершина – правое поддерево
c) Правое поддерево – вершина – левое поддерево
d) Левое поддерево – правое под-дерево – вершина

24. Закончите фразу: «Если вершина бинарного дерева имеет одного непосредственного потомка, то этот потомок…
a) является левой дочерней вершиной»
b) является правой дочерней вершиной»

c) может быть как левой, так и правой дочерней вершиной»
d) не является ни левой, ни правой дочерней вершиной»

25. При обходе дерева слева направо получаем последовательность…
a) отсортированную по убыванию;
b) неотсортированную;
c) отсортированную по возрастанию.

26. При обходе дерева слева направо его элемент заносится в массив…
a) при втором заходе в элемент ;
b) при первом заходе в элемент;
c) при третьем заходе в элемент.

27. Элемент дерева, который не ссылается на другие, называется
a) корнем
b) листом
c) узлом
d) промежуточным

28. Элемент дерева, на который не ссылаются другие, называется
a) корнем
b) листом
c) узлом
d) промежуточным

29. Элемент дерева, который имеет предка и потомков, называется
a) корнем
b) листом
c) узлом
d) промежуточным

30. Высотой дерева называется
a) максимальное количество узлов
b) максимальное количество связей
c) максимальное количество листьев
d) максимальная длина пути от корня до листа

31. Степенью дерева называется
a) максимальная степень всех узлов
b) максимальное количество уровней его узлов
c) максимальное количество узлов
d) максимальное количество связей
e) максимальное количество листьев

32. Дерево называется бинарным, если
a) количество узлов может быть либо пустым, либо состоять из корня с двумя другими бинарными поддеревьями
b) каждый узел имеет не менее двух предков
c) от корня до листа не более двух уровней
d) от корня до листа не менее двух уровней

33. Бинарное дерево можно представить
a) с помощью указателей
b) с помощью массивов
c) с помощью индексов
d) правильного ответа нет

34. Как называются предки узла, имеющие уровень на единицу меньше уровня самого узла
a) детьми
b) родителями
c) братьями

35. Укажите правильное продолжение определения: «Идеально сбалансированным бинарным деревом называется дерево, в котором…
a) каждая нетерминальная вершина имеет ровно два непосредственных потомка»
b) для каждой вершины количество вершин в ее левом и правом поддереве отличается не более чем на 1»
c) для каждой вершины количество вершин в ее левом и правом поддереве совпадает»
d) каждая вершина имеет не более одного непосредственного потомка»

36. Укажите, какие действия требуются для того, чтобы удалить из бинарного дерева, состоящего из трех элементов ,все вершины, кроме корня Root.
a) Положить Root->Left и Root->Right равными NULL
b) Вызвать оператор delete для Root->Left и Root->Right
c) Положить Root->Left и Root->Right равными NULL, после чего оператор delete для Root->Left и Root->Right
d) Вызвать процедуру оператор delete для Root->Left и Root->Right, после чего положить Root->Left и Root->Right равными NULL

37. Укажите верное утверждение.
a) При создании непустого дерева всегда используется рекурсия и может использоваться оператор new
b) При создании непустого дерева всегда используется рекурсия и всегда используется оператор new
c) При создании непустого дерева может использоваться рекурсия и всегда используется оператор new
d) При создании непустого дерева может использоваться рекурсия и может использоваться оператор new

38. Укажите правильное продолжение определения: «Полным бинарным деревом называется дерево глубины L, в котором…
a) число вершин равно 2L»
b) число листьев равно L»
c) число вершин равно 2L+1»
d) число листьев равно 2L»

39. Укажите, какие действия требуются для перемены местами левого и правого потомка вершины P.
a) Поменять значения полей P->Left и P->Right
b) Поменять значения полей P->Left->Data и P->Right->Data
c) Поменять значения полей P->Left->Data и P->Right->Data, после чего поменять значения полей полей P->Left и P->Right
d) Поменять значения полей P->Left и P->Right, после чего поменять значения полей P->Left->Data и P->Right->Data

40. Укажите правильный вариант алгоритма создания идеально сбалансированного дерева с N вершинами.
a) Алгоритм создает корень дерева, после чего рекурсивно вызывается для создания левого и правого поддерева с N / 2 вершинами
b) Алгоритм создает корень дерева, после чего рекурсивно вызывается для создания левого и правого поддерева с N – 1 – N / 2 вершинами

c) Алгоритм создает корень дерева, после чего рекурсивно вызывается для создания левого поддерева с N / 2 вершинами и правого поддерева с N – 1 – N / 2 вершинами
d) Алгоритм создает корень дерева, после чего рекурсивно вызывается для создания левого поддерева с N / 2 вершинами и правого поддерева с N – N / 2 вершинами
41. Укажите, какому значению пропорционально число операций в алгоритме сортировки деревом, примененном к набору из N элементов.
a) N2
b) N log2 N
c) N log10 N
d) N3/2

42. Укажите неправильное продолжение для следующего начала фразы: «Если бинарное дерево является деревом поиска, то…
a) значение каждой его вершины не больше значений вершин ее правого поддерева»
b) максимальное значение в левом поддереве любой вершины не превосходит значение этой вершины»
c) при обходе его вершин в инфиксном порядке значения вершин образуют неубывающую последовательность»
d) при обходе его вершин в префиксном порядке значения вершин образуют невозрастающую последовательность»

ДЕ-3.
1. В чем состоит основной принцип кодирования методом Хаффмана?
a) Каждый символ кодируется бинарной последовательностью одинаковой длины;
b) Символы кодируются избыточным кодом постоянной длины с проверкой на четность;
c) Символы, встречающиеся чаще, кодировать бинарной последовательностью меньшей длины, а символы, встречающиеся реже, - бинарной последовательностью большей длины.

2. Метод кодирования Хаффмана является двухпроходным. Что это значит?
a) С помощью этого метода можно как кодировать информацию, так и декодировать;
b) На первом проходе строится алфавит и генерируются коды. На втором проходе происходит непосредственно кодирование;
c) 3На первом проходе строится код для часто встречающихся символов. На втором – для редко встречающихся символов.

3. Каким образом считывается код с бинарного дерева Хаффмана?
a) От корня бинарного дерева к его узлам;
b) От узлов бинарного дерева к его корню;
c) Можно считывать как в одном направлении, так и в другом.

ДЕ-4.
1. Граф – это
a) Нелинейная структура данных, реализующая отношение «многие ко многим»;
b) Линейная структура данных, реализующая отношение «многие ко многим»;
c) Нелинейная структура данных, реализующая отношение «многие к одному»;
d) Нелинейная структура данных, реализующая отношение «один ко многим»;
e) Линейная структура данных, реализующая отношение «один ко многим».

2. Узлам (или вершинам) графа можно сопоставить:
a) отношения между объектами;
b) объекты;
c) связи
d) типы отношений
e) множества

3. Рёбрам графа можно сопоставить:

a) связи
b) типы отношений
c) множества
d) объекты;
e) отношения между объектами;

4. Граф, содержащий только ребра, называется.
a) ориентированным
b) неориентированным
c) простым
d) смешанным

5. Граф, содержащий только дуги, называется.
a) ориентированным
b) неориентированным
c) простым
d) смешанным

6. Граф, содержащий дуги и ребра, называется.
a) ориентированным
b) неориентированным
c) простым
d) смешанным

7. Есть несколько способов представления графа в ЭВМ. Какой из способов приведенных ниже не относится к ним.
a) матрица инциденций;
b) матрица смежности;
c) список ребер;
d) массив инцидентности.

8. Если последовательность вершин v0, v1, …vp определяет путь в графе G, то его длина определяется:
a) ;
b) ;
c) ;
d) .

9. Каким образом осуществляется алгоритм нахождения кратчайшего пути от вершины s до вершины t
a) нахождение пути от вершины s до всех вершин графа
b) нахождение пути от вершины s до заданной вершины графа
c) нахождение кратчайших путей от вершины s до всех вершин графа
d) нахождение кратчайшего пути от вершины s до вершины t графа
e) нахождение всех путей от каждой вершины до всех вершин графа

10. Как определяется длина пути дерева
a) как сумма длин путей всех его узлов
b) как количество ребер от узла до вершины
c) как количество ребер от листа до вершины
d) как максимальное количество ребер
e) как максимальное количество листьев
f) как длина самого длинного пути от ближнего узла до какого-либо листа

11. Суть алгоритма Дейкстры - нахождения кратчайшего пути от вершины s до вершины t заключается

a) вычислении верхних ограничений d[v] в матрице весов дуг a[u,v] для u, v
b) вычислении верхних ограничений d[v]
c) вычислении верхних ограничений в матрице весов дуг a[u,v]
d) вычислении нижних ограничений d[v] в матрице весов дуг a[u,v] для u, v

12. Граф, содержащий только ребра, называется
a) ориентированным
b) неориентированным
c) простым
d) связным

13. Граф, содержащий только дуги, называется
a) ориентированным
b) неориентированным
c) простым
d) связным

14. Граф, содержащий ребра и дуги, называется
a) неориентированным
b) простым
c) смешанным
d) связным

15. Путь(цикл), который содержит все ребра графа только один раз, называется
a) Эйлеровым
b) Гамильтоновым
c) декартовым
d) замкнутым

ДЕ-5.

1. Как называется сортировка, происходящая в оперативной памяти?
a) сортировка таблицы адресов;
b) полная сортировка;
c) сортировка прямым включением;
d) внутренняя сортировка ;
e) внешняя сортировка.

2. Как можно сократить затраты машинного времени при сортировке большого объёма данных?
a) производить сортировку в таблице адресов ключей ;
b) производить сортировку на более мощном компьютере;
c) разбить данные на более мелкие порции и сортировать их.

3. Существуют следующие методы сортировки. Найдите ошибку.
a) строгие;
b) улучшенные;
c) динамические.

Д-6.

1. Где эффективен линейный поиск?
a) в списке;
b) в массиве;
c) в массиве и в списке.

2. Какой поиск эффективнее?
a) линейный;
b) бинарный;

a) связи
b) типы отношений
c) множества
d) объекты;
e) отношения между объектами;

4. Граф, содержащий только ребра, называется.
a) ориентированным
b) неориентированным
c) простым
d) смешанным

5. Граф, содержащий только дуги, называется.
a) ориентированным
b) неориентированным
c) простым
d) смешанным

6. Граф, содержащий дуги и ребра, называется.
a) ориентированным
b) неориентированным
c) простым
d) смешанным

7. Есть несколько способов представления графа в ЭВМ. Какой из способов приведенных ниже не относится к ним.
a) матрица инциденций;
b) матрица смежности;
c) список ребер;
d) массив инцидентности.

8. Если последовательность вершин v0, v1, …vp определяет путь в графе G, то его длина определяется:
a) ;
b) ;
c) ;
d) .

9. Каким образом осуществляется алгоритм нахождения кратчайшего пути от вершины s до вершины t
a) нахождение пути от вершины s до всех вершин графа
b) нахождение пути от вершины s до заданной вершины графа
c) нахождение кратчайших путей от вершины s до всех вершин графа
d) нахождение кратчайшего пути от вершины s до вершины t графа
e) нахождение всех путей от каждой вершины до всех вершин графа

10. Как определяется длина пути дерева
a) как сумма длин путей всех его узлов
b) как количество ребер от узла до вершины
c) как количество ребер от листа до вершины
d) как максимальное количество ребер
e) как максимальное количество листьев
f) как длина самого длинного пути от ближнего узла до какого-либо листа

11. Суть алгоритма Дейкстры - нахождения кратчайшего пути от вершины s до вершины t заключается

c) без разницы.

3. В чём суть бинарного поиска?
a) нахождение элемента массива x путём деления массива пополам каждый раз, пока элемент не найден ;
b) нахождение элемента x путём обхода массива;
c) нахождение элемента массива x путём деления массива.

4. Как расположены элементы в массиве бинарного поиска?
a) по возрастанию;
b) хаотично;
c) по убыванию.

5. В чём суть линейного поиска?
a) производится последовательный просмотр от начала до конца и обратно через 2 элемента;
b) производится последовательный просмотр элементов от середины таблицы;
c) производится последовательный просмотр каждого элемента.

6. Что такое уникальный ключ?
a) если разность значений двух данных равна ключу;
b) если сумма значений двух данных равна ключу;
c) если в таблице есть только одно данное с таким ключом.

7. В чём состоит назначение поиска?
a) среди массива данных найти те данные, которые соответствуют заданному аргументу ;
b) определить, что данных в массиве нет;
c) с помощью данных найти аргумент.

8. Какой метод поиска представлен в следующем фрагменте do{ i++; while (A[i]!=X) && (i!=N);
a) последовательный
b) двоичный
c) восходящий
d) нисходящий
e) смешанный

9. Какой метод поиска представлен в следующем фрагменте do k=(i+j)/ 2; if(x>A[k]) i=k+1; else j=k-1;
while (A[k]!=x) && (i<=j);
a) последовательный
b) бинарный
c) восходящий
d) нисходящий
e) смешанный

10. Реализация поиска в линейном списке выглядит следующим образом
a) while (!P ||(P->KEY!=X)) P=P->NEXT;
b) while (!P) P=P->NEXT
c) while && (P->KEY!=X) P=P->NEXT
d) while (!PL)&& (P->KEY!=X) P=P->NEXT
e) while (!P P->KEY!=X) P=P->NEXT

11. В графах общая идея поиска в глубину состоит в следующем:
a) Поиск начинаем с некоторой фиксированной вершины v0. Затем выбираем произвольную вершину u, смежную с v0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем

продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если v=v0, то поиск закончен);

b) Поиск начинаем с некоторой фиксированной вершины v0. Затем выбираем произвольную вершину u, смежную с v0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если v=u, то поиск закончен);

c) Поиск начинаем с некоторой фиксированной вершины v0. Затем выбираем произвольную вершину u, смежную с v0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если v=v0, то поиск закончен).

12. Стандартным способом устранения рекурсии при поиске в глубину является использование:

a) массива;
b) очереди;
c) стека;
d) циклического списка.

13. При поиске в ширину используется:

a) массив;
b) очередь;
c) стек;
d) циклический список.

14. Имеется неупорядоченный массив целых чисел из N элементов. Сколько операций сравнения потребуется для установления факта отсутствия искомых данных в этом массиве, используя алгоритм последовательного поиска?

a) N
b) 1
c) N/2
d) 0
e) N-1

15. Имеется неупорядоченный массив целых чисел из N элементов. Сколько операций сравнения потребуется для нахождения искомого ключа, если он находится в конце массива, используя алгоритм последовательного поиска?

a) N
b) 1
c) N/2
d) 0
e) N-1

Ключи к тестам.

№	Ответ
ДЕ-1	
1	A
2	B
3	C
ДЕ-2	
1	A
2	C
3	C
4	A

5	C
6	B
7	C
8	B
9	C
10	b
11	a
12	c
13	C
14	C

15	A
16	A
17	B
18	C
19	A
20	A
21	B
22	D
23	B
24	C
25	B
26	A
27	B
28	A
29	D
30	D
31	A
32	A
33	a,b
34	B
35	B
36	D
37	C
38	D
39	A
40	C
41	B
42	D
ДЕ-3	
1	C
2	B
3	A
ДЕ-4	
1	A

2	B
3	E
4	B
5	A
6	D
7	d
8	a
9	a
10	a
11	a
12	b
13	a
14	c
15	A
ДЕ-5	
1	d
2	a
3	c
ДЕ-6	
1	c
2	b
3	a
4	a
5	c
6	c
7	a
8	a
9	c
10	a
11	a
12	c
13	b
14	a
15	a

«УТВЕРЖДАЮ»
зав. кафедрой информатики и программной инженерии,
доцент_____ Л.А. Тягульская
«21»_____09_____2023 г.

Вопросы к зачету
по дисциплине «Типы и структуры данных»
для студентов II курса
направления подготовки «Программная инженерия»
профиль подготовки «Разработка программно-информационных систем»,
III семестр

1. Указатели.
2. Динамические структуры данных.
3. Линейный однонаправленный список.
4. Линейный двунаправленный список.
5. Линейный однонаправленный список. Вставка и удаление элемента.
6. Линейный двунаправленный список. Вставка и удаление элемента.
7. Стеки. Реализация стеков.
8. Очереди. Реализация очереди.
9. Операции, проводимые с элементами линейного списка.
10. Операции, проводимые с элементами очереди.
11. Операции, проводимые с элементами стека.
12. Деревья. Основные понятия.
13. Реализация деревьев с помощью массива.
14. Алгоритмы обходов бинарного дерева.
15. Добавление элемента в двоичное дерево поиска.
16. Алгоритм Хаффмана.
17. Поиск элемента в двоичном дереве поиска.
18. Балансировка деревьев
19. Алгоритмы представления графа.
20. Обходы в графах.
21. Поиск кратчайших путей. Алгоритм Дейкстры.
22. Алгоритмы сортировок: выбором, вставками, пузырьковая, быстрая, слиянием, пирамидальная.

Экзаменатор, преподаватель _____ О.М. Нагаевский

«УТВЕРЖДАЮ»
зав. кафедрой информатики и программной инженерии,
доцент_____ Л.А. Тягульская
«21»_____09_____2023 г.

Практические задания к экзамену по дисциплине «Типы и структуры данных» для студентов II курса направления «Программная инженерия» профиля подготовки «Разработка программно-информационных систем», III семестр

Практическое задание № 1

Пример программы:

```
#include <iostream.h>
void main()
{
        int a=l, b=2;
        int *c=&b, *d=&a;
        cout« a «' '« b«' '«*c«' '« d«' '« c«' '« *d«' '« "\n";
}
```

Задача:

Используя стек, элементами которого являются целые числа, написать программу, которая находит количество близнецов, находящихся в стеке. Близнецы - это два простых числа, разность которых равна 2.

Практическое задание № 2

Пример программы:

```
# include <stdio.h>
void main ( void )
{
int count, *point_count;
count = 5;
point_count = &count;
*point_count = 10;
}
```

Задача:

Используя стек, элементами которого являются целые числа, написать программу, которая удаляет из стека все элементы, кратные 4 (использовать для промежуточного хранения элементов стека однонаправленный список с заглавным звеном).

Практическое задание № 3

Пример программы:

```
struct L
{
int data;
```

```
L *next;
} *Top;
void main()
{
L *p;
...
p=Top;
 Top=Top->next;
 delete p;
...
}
```

Задача:

Напишите программу добавления в конец очереди Q элемента X из ее начала, причем при формировании очереди учтите, что она может содержать не более K звеньев (переполнение очереди) и не может быть пустой (опустошение очереди)

Практическое задание № 4

Пример программы:

```
struct L
{
int data;
L *next;
} *Top;
void main()
{
L *p;
...
 p=Top;
while(p!=0)
 {
cout<<p->data<<" ";
 p=p->next;
 }
...
}
```

Задача:

Используя непустой однонаправленный список, элементами которого являются целые числа, написать программу, которая должна перенести в конец этого списка его первый элемент.

Практическое задание № 5

Пример программы:

```
struct L
{
int data;
L *next;
} *Top;

 void main()
 {
 L *p;
 ...
  p=Top;
  while(p!=0)
```

```
 {
 Top=Top->next;
 delete p;
  p=Top;
 }
 ...
}
```

Задача:

Используя непустой однонаправленный список, элементами которого являются вещественные числа, написать программу, которая по списку строит два новых списка: L1 - из положительных элементов списка P, L2 - из отрицательных элементов списка P.

Практическое задание № 6

Пример программы:

```
struct E
{
char data;
E * Next;
}* Head,* Tail;

void d(char data)
{ E * temp = new E;
  temp->data = data;
 temp->Next = NULL;
  if(Head!=NULL){
Tail->Next=temp;
Tail = temp;}
 else{ Head=Tail=temp;}
}
```

Задача:

Реализовать стек, с проверкой переполнения и удаления.

Практическое задание № 7

Пример программы:

```
struct element
{element *prev;
int info;
element *next;
};
element *head;
element *tail;

void turn (int x)
{element *temp;
temp=new element;
temp->info=x;
temp->next=0;
temp->prev=tail;
if(head==0)
head=tail=temp;
else
{tail->prev=temp;
tail=temp;
}
```

```
};
```
Задача:

Реализовать стек, таким образом, чтоб в нем были запрещены повторяющиеся элементы.

Практическое задание № 8

Пример программы:

```
void  Scan(bt *p, int le)
{
if (p==NULL) return;
Scan(p->left,le+1);
Cout<<le<<p->val;
Scan(p->right,le+1);
}
```

Задача:

Написать программу, которая проверяет, упорядочены ли элементы списка по алфавиту.

Практическое задание № 9

Пример программы:

```
long count ()
{   if  (top== 0)    return 0;
    int c;
   list *temp=top;
for(c=0;(temp);c++,temp=temp->next);
 return c;
}
```

Задача:

Задан текст, состоящий из строк, разделенных пробелом и оканчивающийся точкой. В списке подсчитать количество вхождений заданного символа в каждую строку текста. Вхождение задавать номером строки и номером позиции в строке.

Практическое задание № 10

Пример программы:

```
struct L
{
int data;
L *next;
} *Top;

void main()
{
L *p;
...
p=Top;
while(p!=0)
 {
cout<<p->data<<" ";
 p=p->next;
 }
...
}
```

Задача:

Дан текстовый файл T, используя стек, напечатать его содержимое в обратном порядке.

Практическое задание № 11

Пример программы:

```
# include < stdio.h>
```

```
void main()
{
char *point_Ch;
unsigned long l,*point_l;
point_l = &l;
point_Ch = (char *)point_l;
}
```

Задача:

Написать программу, которая проверяет, входит ли вершина, содержащая информационное поле E, в правое или левое поддерево заданного бинарного дерева.

Практическое задание № 12

Пример программы:

```
void array_1(int ,int *);
void main()
{
int array[20];
array_1 (20,array);
}
void array_1(int n,int *array)
{
int i;
for (i = 0 ; i < n; i++)
*(array + i) = 1;
}
```

Задача:

Используя структуру данных очередь, переписать содержимое текстового файла F в текстовый файл G, перенося при этом в конец каждой строки все входящие в нее цифры (с сохранением исходного взаимного порядка, как среди цифр, так и среди остальных литер строки).

Практическое задание № 13

Пример программы:

```
void main()
{
int a,*point_1,**point_2,***point_3;
a = 5;
point_1 = &a;
point_2 = &point_1;
point_3 = &point_2;
***point_3 = 10;
}
```

Задача:

Написать программу, которая определяет количество вхождений элемента E в информационные поля вершин левого поддерева заданного бинарного дерева.

Практическое задание № 14

Пример программы:

```
struct L
{
int data;
L *next;
} *Top;
```

```
void main()
{
L *p;
...
if (!p)
   {
    p=new L;
    p->next=0;
    cin>>p->data;
    Top=p;
   }
 else
   {
    L *t;
    t=new L;
    cin>>t->data;
    t->next=Top;
    Top=t;
   }
}
...
}
```

Задача:

Построить очередь из файла, содержащего вещественные числа.

Практическое задание № 15

Пример программы:

```
struct L
{
int data;
L *next;
} *Top;
void main()
{
L *p;
...
 p=Top;
while(p!=0)
 {
cout<<p->data<<" ";
 p=p->next;
 }...}
```

Задача:

Используя очередь, содержащую целые числа, вычислить количество простых чисел, содержащихся в очереди.

Практическое задание № 16

Пример программы:

```
struct E
{
char data;
E * Next;
}* Head,* Tail;

 void d(char data)
```

```
{
 E * temp = new E;
 temp->data = data;
 temp->Next = NULL;
 if(Head!=NULL){
Tail->Next=temp;
Tail = temp;
 } else{
  Head=Tail=temp;
 }
}
```

Задача:

Построить бинарное дерево для заданного множества целых чисел и занумеровать его вершины в соответствии с левосторонним обходом.

Практическое задание № 17

Пример программы:

```
void  Scan(bt *p, int le)
{
if (p==NULL) return;
Scan(p->left,le+1);
Cout<<le<<p->val;
Scan(p->right,le+1);
}
```

Задача:

Используя однонаправленный список, элементами которого являются целые числа, написать программу, которая из списка удаляет все вхождения данного элемента.

Практическое задание № 18

Пример программы:

```
void main()
{
int a,*point_1,**point_2,***point_3;
a = 5;
point_1 = &a;
point_2 = &point_1;
point_3 = &point_2;
***point_3 = 10;
}
```

Задача:

Используя однонаправленный список, элементами которого являются символы, написать программу, которая меняет местами первый и последний элементы непустого списка L.

Практическое задание № 19

Пример программы:

```
void array_1(int ,int *);
void main()
{
int array[20];
array_1 (20,array);
}
void array_1(int n,int *array)
{
int i;
```

```
for (i = 0 ; i < n; i++)
*(array + i) = 1;
}
```

Задача:

Используя очередь, содержащую целые числа, вычислить количество чисел, свободных от квадратов и содержащихся в очереди. Число называется свободным от квадратов, если оно не делится ни на один квадрат простого числа.

Практическое задание № 20

Пример программы:

```
# include < stdio.h>
void main()
{
char *point_Ch;
unsigned long l,*point_l;
point_l = &l;
point_Ch = (char *)point_l;
}
```

Задача:

Используя очередь, содержащую целые числа, сформировать новую очередь P, состоящую из элементов очереди Q, кратных числу 3.

Практическое задание № 21

Пример программы:

```
# include <stdio.h>
void main ( void )
{
int count, *point_count;
count = 5;
point_count = &count;
*point_count = 10;
}
```

Задача:

Написать программу, которая определяет, входит ли вершина, содержащая информационное поле E, в заданное бинарное дерево дважды.

Практическое задание № 22

Пример программы:

```
struct L
{
int data;
L *next;
} *Top;

void main()
{
L *p;
...
p=Top;
while(p!=0)
 {
cout<<p->data<<" ";
 p=p->next;
 }
...
```

```
}
```
Задача:

Написать программу, которая определяет количество вхождений элемента E в информационные поля вершин правого поддерева заданного бинарного дерева.

Практическое задание № 23

Пример программы:

```
struct element
{element *prev;
int info;
element *next;
};
element *head;
element *tail;

void turn (int x)
{element *temp;
temp=new element;
temp->info=x;
temp->next=0;
temp->prev=tail;
if(head==0)
head=tail=temp;
else
{tail->prev=temp;
tail=temp;
}
};
```

Задача:

Используя однонаправленные списки L1 и L2, элементами которых являются целые числа, написать программу, которая в список L1, элементы которого упорядочены по неубыванию, вставляет первый элемент списка L2 так, чтобы сохранилась упорядоченность.

Практическое задание № 24

Пример программы:

```
# include < stdio.h>
void main()
{
char *point_Ch;
unsigned long l,*point_l;
point_l = &l;
point_Ch = (char *)point_l;
}
```

Задача:

Построить бинарное дерево для заданного множества целых чисел и занумеровать его вершины в соответствии с правосторонним обходом.

Экзаменатор, преподаватель _____ О.М. Нагаевский