**Структуры и Алгоритмы обработки данных**

**1 Введение в предмет**

1. **В чем заключаются основные различия между непрерывной и**

**дискретной информацией?**

Человеком информация воспринимается в виде информационных полей, математически которые можно описать в виде функции y = f (x,t) , где t – время; x – точка, в которой измеряется поле; y – величина поля в этой точке.

Величины, входящие в эту формулу, принимают скалярные значения в

определенном диапазоне. Их значения могут быть настолько близки друг к другу, что практически они не различимы, но на самом деле отличаются

друг от друга, а их множество является бесконечным. Такие величины

называются непрерывными, а информация, которую они несут, является

**непрерывной, или аналоговой**.

Если же непрерывную величину поделить на минимальные отрезки,

которые отражают изменение этой величины во времени, то мы получим

так называемую дискретную величину и, соответственно, **дискретную**

**информацию**, которую называют также **цифровой информацией**.

1. **Каким образом можно привести непрерывную информацию к**

**дискретному виду**?

Для восприятия человеком и обработки информации на ЭВМ любая

непрерывная информация может быть приведена к дискретной путем ее

аппроксимации.

1. **Дайте определение понятию ≪алфавит данных ЭВМ≫.**

Конечное множество символов, которыми кодируется информация,

называется **Алфавитом**.

1. **Приведите определение и классификацию типов данных.**

**Тип данных** определяет множество значений, которые могут принимать

данные, множество допустимых операций над данными и способы

организации хранения данных в памяти ЭВМ.

В языке программирования Си типы данных поделены на две группы:

1. Простые данные – это целые, вещественные числа, символы, указатели

на какой-то другой тип (адреса объектов какого-либо типа в памяти). Веще-

ственные числа имеют дробную часть, у целых чисел эта часть отсутствует.

2. Структурированные типы данных представляют собой данные, образо-

ванные на основе других типов, которыми могут быть другие (вложенные)

структуры или простые типы данных.

1. **Какие простые типы данных есть в языке программирования Си, сколько**

**байтов занимают эти типы в памяти?**

В языке Си существуют следующие простые типы данных:

• char – символьный; 1байт;

• int – целый; 4байт;

• float – вещественный; 4байт;

• double – вещественный тип двойной точности; 8байт;

• void – тип, не имеющий значения (используется для описания функ-

ций). 1байт;

1. **Что такое модификатор доступа к типу данных в языке Си?**

в языке Си есть следующие модификаторы типов:

• unsigned – беззнаковый;

• signed – со знаком;

• short – короткий;

• long – длинный.

Модификаторы типа записываются перед именем типа, например: long int. Если после модификатора имя типа опущено, то считается, что этим типом является int.

1. **Для чего используется тип void в языке Си?**

Тип void (не имеющий значения) используется для нейтрализации

значений объекта, например для объявления функции, не возвращаю-

щей никаких значений.

**2 Структурированные типы данных**

1. **Какой тип данных называется структурированным?**

**Структурированный тип данных** – это множество элементов данных,

объединенных одним именем.

1. **Какие виды структурированных типов бывают?**

• **массивы**. Доступ к элементу массива осуществляется через индекс

(порядковый номер, начиная с 0) элемента;

• **строки** (массивы символьных данных). Доступ к элементу строки

осуществляется аналогичным доступу к элементу массива способом;

• **структуры** (записи в языке Паскаль). Доступ к структуре осуществля-

ется посредством имени структурной переменной, более подробно это

будет рассмотрено в п. 2.2;

• **объединения**. Доступ осуществляется аналогичным доступу к струк-

туре образом;

• **перечисления**. Доступ по имени переменной перечислимого типа.

1. **Что такое массив? Какие способы задания массивов существуют в**

**языке Си?**

- Массив состоит из элементов одного и того же типа.

Массивы определяются так же, как и переменные:

int a[100];

char b[20];

float d[50];

char а[10][20];

int a[5] = {0, 1, 2, 3, 4};

1. **Что такое указатель? Как в языке Си задается массив с помощью ука-**

**зателей?**

**Указатель** – это переменная, которая содержит адрес области памяти, в

которой хранятся данные определенного типа (простого или

структурированного).

В языке Си допускаются массивы указателей, которые определяются,

например, следующим образом: char \*m[5];. Здесь m[5] – массив, содер-

жащий адреса элементов типа char

1. **Какие существуют способы инициализации указателя?**

1. Присваивание указателю адреса существующего объекта:

• с помощью операции получения адреса:

int а = 5; // переменная целого типа

int\* р = &а; //в указатель p записывается адрес переменной а

• с помощью значения другого инициализированного указателя:

int\* r = р; //p – уже инициированный указатель

• с помощью имени массива или функции, которые трактуются как

адрес:

int b[10]; // массив

int \* t = b; // присваивание адреса начала массива

void f (int а ) { /\* ... \* / } // определение функции

void (\*pf) (int); // указатель на функцию

pf = f; // присваивание адреса функции

2. Присваивание пустого значения:

int\* SUXX = NULL; [4].

1. **В чем заключается отличие динамического массива от статического?**

**Динамический массив** – это массив, размер которого в памяти может изменяться в течение работы программы, причем неоднократно.

**Динамический массив** всегда использует участок свободной

памяти, который также может называться кучей. Следует помнить,

что любой указатель может быть так же проиндексирован, как и

массив.

**Статический массив** определяется на этапе компиляции и его размеры не меняются.

1. **Что делает функция malloc()?**

Функция malloc() выделяет память и возвращает указатель типа void\* на

начало выделенной памяти.

1. **Приведите алгоритм выделения памяти под двумерный**

**динамический массив.**

m – количество строк

n – количество столбцов

mas = (int\*\*) malloc(sizeof(int\*)\*m);

for(i=0;i<m;i++)

mas[i]=(int\*) malloc(sizeof(int)\*n);

1. **Что такое строка в языке Си? Объявление строки?**

В качестве строк в языке Си используются массивы символов, для которых определены операции, отличные от операций, производимых над массивами других типов данных.

В языке Си для представления строк используется два формата:

1. формат ANSI;

Формат ANSI устанавливает, что значением первой позиции в строке яв-

ляется ее длина, а затем следуют сами символы строки. Например, представление строки ≪Это строка!≫ будет следующим: 'Э''т''о'' ''с''т''р''о''к''а''!' .

1. строки с завершающим нулем (используется в Си);

В строках с завершающим нулем значащие символы строки

Указываются с первой позиции, а признаком завершения строки

является значение ноль. Представление рассмотренной ранее строки в

этом формате имеет вид: 'Э''т''о'' ''с''т''р''о''к''а''!'0.

1. **Что в языке Си обозначает символ ≪\0≫?**

Строки с завершающим нулем (используется в Си);

В строках с завершающим нулем значащие символы строки

Указываются с первой позиции, а признаком завершения строки

является значение ноль. Представление рассмотренной ранее строки в

этом формате имеет вид: 'Э''т''о'' ''с''т''р''о''к''а''!'0.

1. **Перечислите функции для работы со строками библиотеки**

**string.h.**

Все библиотечные функции, предназначенные для работы со строками,

можно разделить на три группы:

1. ввод и вывод строк;

2) преобразование строк;

3) обработка строк. string.h

Вычисление длинны строки:

size\_t strlen(const char \*string);

Копирование:

char \* strcpy(char \* restrict dst, const char \* restrict src);

char \* strncpy(char \* restrict dst, const char \* restrict src, size\_t num);

Сравнение:

int strcmp(const char \*string1, const char \*string2);

int strncmp(const char \*string1, const char \*string2,size\_tnum);

Объединение и конкатенация:

char \* strcat(char \* restrict dst, const char \* restrict src);

char \* strncat(char \* restrict dst, const char \* restrict src, size\_t num);

Функции поиска символа в строке:

char \* strchr(const char \*string, int c);

char \* strrchr(const char \*string, int c);

Функция поиска строки в строке:

char \* strstr(const char \*str, const char \*substr);

Функция поиска первого символа в строке из заданного набора символов:

size\_t strcspn(const char \*str, const char \*charset);

Функция поиска следующего литерала в строке:

char \* strtok(char \* restrict string, const char \* restrict charset);

1. **Расскажите о доступе к членам структуры. В каких случаях для до-**

**ступа используется оператор ≪.≫, а в каких ≪->≫?**

Структура – это одна или несколько переменных (возможно, различных типов),

которые для удобства работы с ними сгруппированы под одним именем.

struct point {

int x;

int y;};

Перечисленные в структуре переменные называются элементами (членами) структуры.

Доступ к отдельному элементу структуры осуществляется посредством

конструкции вида: имя\_структуры.элемент.

Если p – указатель на структуру, то р-> элемент структуры есть ее отдельный

элемент. (Оператор -> состоит из знака ≪-≫, за которым сразу следует

знак ≪>≫.)

1. **Опишите любую предметную область с помощью структур.**

typedef struct Stud /\*описание структурированного типа Stud,

обозначающего студента\*/

{

int grnum;//номер группы студента

char fam[20], name[10], sname[20];/\*фамилия, имя, отче-

ство студента\*/

} stud; //псевдоним stud структурированного типа Stud

1. **Как происходит переименование типов данных в языке Си?**

используется оператор typedef переименования типа, имеющий общий вид:

typedef тип новое\_имя [ размерность ];

где размерность может быть не задана

typedef struct Stud {

int grnum;//номер группы студента

char fam[20], name[10], sname[20];/\*фамилия, имя, отче-

ство студента\*/

} stud;

В данном примере переименовываемым типом является тип struct

Stud (имя начинается с прописной буквы S, хотя может быть абсолютно лю-

бым, но отличным от нового имени), а новым именем структуры struct

Stud является stud, написанное в конце описания структуры, согласно син-

таксису оператора typedef.

1. **Что Вы знаете о вычислении размера простого или структурирован-**

**ного типа данных в языке Си?**

Выражения sizeof объект и sizeof (имя типа) выдают целые значения,

равные размеру указанного объекта или типа в байтах.

1. **Как происходит чтение и запись массива структур в двоичный файл?**

Для чтения и записи данных из двоичных файлов в языке Си существуют

функции fread и fwrite. Исходя из вышеизложенных принципов, функции

двоичного ввода-вывода fread и fwrite переносят содержимое памяти в

двоичный файл ≪прозрачно≫, т. е. байт в байт без каких либо

преобразований.

Функции используются для перенесения данных из файла в память

программы (чтение) и обратно (запись).

int fread (void \*buf, int size, int nrec, FILE \*fd) – чтение

(ввод) данных из файла;

int fwrite (void \*buf, int size, int nrec, FILE \*fd) – запись

(вывод) данных в файл.

1. **Опишите тип данных ≪объединение≫.**

Объединение в Си может содержать объявления полей разных типов, кроме

типа самого данного объединения.

Особенностью объединения в Си является то, что все поля объединения имеют одинаковый адрес в памяти. Это приводит к тому, что в данный момент можно

определить значение только одного поля объединения в Си.

union myUnion

{ int nInt;

double dDouble;

char cChar;

char \*szString;

};

union myUnion: union myUnion uUnion = {150}; ,

инициализировать можно только первую переменную объединения в Си.

1. **Опишите тип данных ≪перечисление≫.**

Перечисление – это конструируемый тип данных, во время описания которого

задается имя типа данных и значения, которые могут принимать переменные

этого типа.

enum <имя\_типа> {<список\_значений>}

<список\_переменных>;

enum day {saturday,sunday=0,monday,

tuesday,wednesday,thursday,

friday

} workday;

enum day nextday;

enum day today=monday;

day vacancy;

3 **Алгоритмы сортировки массивов**

1. **Что такое сортировка и для чего она применяется?**

Сортировка – это упорядочивание набора однотипных данных по возрастанию

или убыванию.

В общем случае сортировку следует понимать как процесс перегруппи-

ровки заданного множества объектов в определенном порядке.

1. **Опишите сортировку методом пузырька.**

Алгоритм попарного сравнения элементов массива в литературе часто

называют ≪методом пузырька≫, проводя аналогию с пузырьком, поднимаю-

щимся со дна бокала с газированной водой. По мере всплывания пузырек стал-

кивается с другими пузырьками и, сливаясь с ними, увеличивается в объеме.

Алгоритм состоит в повторяющихся проходах по сортируемому массиву.

За каждый проход элементы последовательно сравниваются попарно, и если

порядок в паре неверный, выполняется обмен элементов. Проходы по массиву

повторяются до тех пор, пока на очередном проходе не окажется, что обмены

больше не нужны, это означает – массив отсортирован.

Пузырьковая сортировка имеет такую особенность: неупорядоченные

элементы на ≪большом≫ конце массива занимают правильные положения за

один проход, но неупорядоченные элементы в начале массива поднимаются на свои места очень медленно.

1. **Опишите сортировку прямым выбором.**

При данной сортировке из массива выбирается элемент с наименьшим

значением и обменивается с первым элементом.

Затем из оставшихся n – 1 элементов снова выбирается элемент с наименьшим

ключом и обменивается со вторым элементом и т. д.

1. Находим минимальное значение в текущей части массива.

2. Производим обмен этого значения со значением на первой

неотсортированной позиции.

3. Далее сортируем хвост массива, исключив из рассмотрения уже

отсортированные элементы.

Несмотря на то что количество сравнений в пузырьковой сор-

тировке и сортировке простым выбором одинаковое, при сортиров-

ке простым выбором количество обменов в среднем случае намного

меньше, чем в пузырьковой сортировке

1. **Расскажите о пирамидальной сортировке.**

Начнем рассмотрение эффективных алгоритмов сортировки (работающих

за O(N log N)) с пирамидальной сортировки, в которой используется идея ≪ку-

чи≫ (т. е. динамической памяти), представляющей собой бинарное дерево (пи-

рамиду.

Все узлы бинарного дерева имеют от 0 до 2 потомков максимум, отсюда и

название – бинарное дерево.

Пирамидальную сортировку следует осуществлять, если

условия задачи таковы, что единственной разрешенной операцией

является ≪проталкивание≫ элемента по куче, либо в случае отсут-

ствия дополнительной памяти

1. **Что Вы знаете об улучшенной пузырьковой сортировке?**

Идея сортировки заключается в последовательных проходах массива. За

каждый проход выполняется обмен соседних элементов (по паре от начала и

конца), если они стоят в неправильном порядке. Проходы выполняются до тех

пор, пока обмены не прекратятся.

Улучшен алгоритм обработки.

1. **Что такое сортировка вставками?**

Хотя этот метод сортировки намного менее эффективен, чем сложные ал-

горитмы (такие как пирамидальная сортировка), у него есть ряд преимуществ:

• прост в реализации;

• эффективен на небольших наборах данных, в последовательностях до

десятков элементов может оказаться лучшим;

• эффективен на наборах данных, которые уже частично отсортирова-

ны;

• это устойчивый алгоритм сортировки (не меняет порядок элементов,

которые уже отсортированы);

• может сортировать массив по мере его получения;

• не использует динамическую память, даже под стек.

На каждом шаге алгоритма выбирается один из элементов входных дан-

ных и вставляется на нужную позицию в уже отсортированной последователь-

ности до тех пор, пока набор входных данных не будет исчерпан.

1. **Что такое сортировка слиянием?**

Сортировка слиянием – вероятно, один из самых простых алгоритмов

сортировки (среди ≪быстрых≫ алгоритмов). Особенностью этого алгоритма яв-

ляется то, что он работает с элементами массива преимущественно последова-

тельно, благодаря чему именно этот алгоритм используется при сортировке в

системах с различными аппаратными ограничениями (например, при сортиров-

ке данных на жёстком диске или даже на магнитной ленте).

Предположим, что у нас есть два отсортированных массива A и B разме-

рами na и nb соответственно, и мы хотим объединить их элементы в один боль-

шой отсортированный массив C размером na + nb. Для этого можно применить

102

процедуру слияния, суть которой заключается в повторяющемся ≪отделении≫

элемента, наименьшего из двух имеющихся в началах исходных массивов, и

присоединении этого элемента к концу результирующего массива. Элементы

мы переносим до тех пор, пока один из исходных массивов не закончится. По-

сле этого оставшийся ≪хвост≫ одного из входных массивов дописывается в ко-

нец результирующего массива.

1. **Расскажите о быстрой сортировке.**

Идея заключается в разделении массива на две части, так чтобы каждый

элемент первой части был меньше элемента второй части. Далее эта операция

выполняется рекурсивно.

1. **Каковы особенности поразрядной сортировки?**

Идея заключается в сортировке разрядов числа по отдельности, а не са-

мих чисел, что позволяет ускорить алгоритм. Для любого варианта заполнения

массива сложность алгоритма (количество операций) составляет O(n), где n –

длина массива.

**4 Динамические структуры данных**

1. **Какие структуры данных называют динамическими?**

Динамические структуры данных – это структуры данных,

память под которые выделяется и освобождается по мере необходимости.

Для решения проблемы адресации динамических структур данных используется метод, называемый динамическим распределением памяти, то есть

память под отдельные элементы выделяется в момент, когда они ≪начинают

существовать≫ в процессе выполнения программы, а не во время компиляции.

Динамическая структура данных характеризуется тем, что

• она не имеет имени (на нее указывает статическая переменная типа

указатель, имеющая имя);

• ей выделяется память в процессе выполнения программы;

• количество элементов структуры может не фиксироваться;

• размерность структуры может меняться в процессе выполнения про-

граммы;

• в процессе выполнения программы может меняться характер взаимо-

связи между элементами структуры.

1. **Что такое указатель?**

Каждой динамической структуре данных соответствует статическая переменная типа указатель (ее значение – адрес этого объекта), посредством которой

осуществляется доступ к динамической структуре

Указатели – это статические величины, поэтому они требуют описания.

1. **Перечислите достоинства и недостатки связного представления**

**данных.**

Для установления связи между элементами динамической структуры используются указатели, через которые устанавливаются явные связи между элементами.

Такое представление данных в памяти называется связным.

Достоинства связного представления данных – в возможности обеспечения

значительной изменчивости структур:

• размер структуры ограничивается только доступным объемом машинной

памяти;

• при изменении логической последовательности элементов структуры

требуется не перемещение данных в памяти, а только коррекция указателей;

• большая гибкость структуры.

Вместе с тем, связное представление не лишено и недостатков, основными из

которых являются следующие:

• на поля, содержащие указатели для связывания элементов друг с другом,

расходуется дополнительная память;

• доступ к элементам связной структуры может быть менее эффективным по

времени.

Порядок работы с динамическими структурами данных:

1. Создать (отвести место в динамической памяти).

2. Работать при помощи указателя.

3. Удалить (освободить занятое структурой место).

1. **Приведите классификацию динамических структур данных.**

К таким структурам относят:

• однонаправленные (односвязные) списки;

• двунаправленные (двусвязные) списки;

• циклические списки;

• стек;

• дек;

• очередь;

• бинарные деревья.

Они отличаются способом связи отдельных элементов и/или допустимы-

ми операциями. Динамическая структура может занимать несмежные участки

оперативной памяти.

1. **Расскажите об инициализации динамических структур данных.**

Каждая компонента любой динамической структуры представляет собой

запись (тип запись используется в языке паскаль и является аналогом типа

структура в языке Си), содержащую по крайней мере два поля: одно поле типа

указатель, а второе – для размещения данных. В общем случае запись может

содержать не один, а несколько указателей и несколько полей данных.

Элемент динамической структуры состоит из двух полей:

• информационного поля (поля данных), в котором содержатся те дан-

ные, для хранения которых и создается структура; в общем случае

информационное поле само является интегрированной структурой –

вектором, массивом, другой динамической структурой и т. п.;

• адресного поля (связного поля), в котором содержатся один или не-

сколько указателей, связывающих данный элемент с другими элемен-

тами структуры.

Объявление элемента динамической структуры данных выглядит следу-

ющим образом:

struct имя\_типа {

информационное поле;

адресное поле;

};

Например:

struct TNode {

int Data;//информационное поле

TNode \*Next;//адресное поле

};

1. **Каковы особенности доступа к данным в динамических структурах?**

Поскольку элементами динамической структуры являются динамические

переменные, то единственным средством доступа к динамическим структурам и их элементам является указатель (адрес) на место их текущего расположения в

памяти.

Указатель содержит адрес определенного объекта в динамической памяти.

Адрес формируется из двух слов: адрес сегмента и смещение. Сам указатель

является статическим объектом и расположен в сегменте данных (описания

переменных программы)

Для обращения к динамической структуре достаточно хранить в памяти

адрес первого элемента структуры. Поскольку каждый элемент динамической

структуры хранит адрес следующего за ним элемента, можно, двигаясь от

начального элемента по адресам, получить доступ к любому элементу данной

структуры.

Доступ к данным в динамических структурах в языке Си осуществляется с

помощью операции ≪стрелка≫ (->), которую называют операцией косвенного

выбора элемента структурного объекта, адресуемого указателем.

Она обеспечивает доступ к элементу структуры через адресующий ее указатель

того же структурного типа.

1. **Что такое динамический список? Приведите классификацию списков.**

Списком называется упорядоченное множество, состоящее из переменного

числа элементов, к которым применимы операции включения и исключения.

Длина списка равна числу элементов, содержащихся в списке, список нулевой

длины называется пустым списком. Списки представляют собой способ

организации структуры данных, при которой элементы некоторого типа

образуют цепочку. Для связывания элементов в списке используют систему

указателей.

Линейные связные списки являются простейшими динамическими струк-

турами данных. Из всего многообразия связанных списков можно выделить

основные:

• однонаправленные (односвязные) списки;

• двунаправленные (двусвязные) списки;

• циклические (кольцевые) списки.

В основном они отличаются видом взаимосвязи элементов и/или допустимыми операциями.

1. **Каковы основные операции над динамическими списками?**

Основные операции, осуществляемые с однонаправленными списками:

• создание списка;

• печать (просмотр) списка;

• вставка элемента в список;

• удаление элемента из списка;

• поиск элемента в списке;

• проверка пустоты списка;

• удаление списка.

1. **Опишите реализацию операций над динамическими списками на**

**языке Си.**

**Дохрена писать**

1. **Дайте определение двунаправленных списков.**

Двунаправленный (двусвязный) список – это структура данных, состоящая из

последовательности элементов, каждый из которых содержит информационную часть и два указателя на соседние элементы (следующий и предыдущий).

При этом два соседних элемента должны содержать взаимные ссылки друг на

друга.

1. **Приведите алгоритм создания двунаправленного списка.**

struct имя\_типа {

<тип> <информационное поле>;

<имя\_типа\*> <адресное поле 1>;

<имя\_типа\*> <адресное поле 2>;

};

1. **Каковы основные операции над двунаправленным списком?**

• создание списка;

• печать (просмотр) списка;

• вставка элемента в список;

• удаление элемента из списка;

• поиск элемента в списке;

• проверка пустоты списка;

• удаление списка.

1. **Распишите алгоритм удаления дубликатов элементов из**

**двунаправленного списка.**

**Дохрена писать**

1. **Дайте определение стека.**

Стек (англ. stack – стопка) – это структура данных, в которой новый элемент всегда записывается в ее начало (вершину) и очередной читаемый элемент также

всегда выбирается из ее начала. В стеках используется дисциплина (метод)

доступа к элементам LIFO (Last Input – First Output, ≪последним пришел – первым

вышел≫).

1. **Расскажите о доступе к элементам стека.**

Стек – это список, у которого доступен только один элемент (одна позиция).

Этот элемент называется вершиной стека.

Взять элемент можно только из вершины стека, добавить элемент можно

только в вершину стека.

Основные операции, производимые со стеком:

• создание стека;

• печать (просмотр) стека;

• добавление элемента в вершину стека;

• извлечение элемента из вершины стека;

• проверка пустоты стека;

• очистка стека.

1. **Что такое обратная польская нотация (ОПН)?**

Математическое выражение в обратной польской нотации (ОПН) – это постфиксная

запись, представляющая собой такую запись арифметического выражения, в

которой сначала записываются операнды, а затем – знак операции.

Например, для выражения a + b \* c ОПН будет записана как

a b c \* +. Здесь операндами операции умножения (\*) будут b и c (два

ближайших операнда), а операндами операции сложения (+) будут а и составной

операнд b c (\*). Эта запись удобна тем, что она не требует скобок.

Например, для выражения (a + b)\*c постфиксная запись будет a b + c\*. В этой

записи не требуется ставить скобки для того, чтобы изменить порядок

вычисления, зависящий от приоритета операций, как в исходном выражении.

1. **Приведите алгоритм перевода математического выражения в ОПН.**

Алгоритм перевода в постфиксную запись обрабатывает исходную

последовательность лексем и строит новую последовательность из тех же

лексем, расположенных в другом порядке. Для перевода математического

выражения в ОПН обычно используют динамическую структуру данных – стек, в который помещаются операнды и промежуточные результаты выполнения

операций (которые также служат операндами в других операциях).

Мы рассмотрим переработанный алгоритм, идея которого предложена

Э. Дейкстра (E. W. Dijkstra). Предположим, что формула состоит из переменных,

двухоперандных операторов +, –, \*, /, а также левой и правой скобок (в самих же постфиксных выражениях скобки мы использовать не будем, поскольку они нам больше не понадобятся).

**5 Деревья (181)**

1. **Из каких элементов состоит древовидная структура данных?**

Дерево – это структура данных, представляющая собой совокупность элементов и отношений, образующих иерархическую структуру этих элементов.

Каждый элемент дерева называется вершиной (узлом) дерева. Вершины

дерева соединены направленными дугами, которые называют ветвями дерева.

Начальный узел дерева называют корнем дерева, ему соответствует нулевой

уровень. Листьями дерева называют вершины, в которые входит одна ветвь и

не выходит ни одной ветви.

Каждое дерево обладает следующими свойствами:

• существует узел, в который не входит ни одна дуга (корень);

• в каждую вершину, кроме корня, входит одна дуга.

1. **Что такое поддерево?**

Поддерево – часть древообразной структуры данных, которая может быть

представлена в виде отдельного дерева.

1. **Что такое степень вершины?**

Степенью вершины в дереве называется количество дуг, которое из нее выходит.

Степень дерева равна максимальной степени вершины, входящей в дерево.

При этом листьями в дереве являются вершины, имеющие степень нуль.

По величине степени дерева различают два типа деревьев:

• двоичные – степень дерева не более двух;

• сильноветвящиеся – степень дерева произвольная.

1. **Что такое бинарное дерево?**

Бинарные деревья являются деревьями со степенью не более двух.

Бинарное (двоичное) дерево – это динамическая структура данных,

представляющая собой дерево, в котором каждая вершина имеет не более двух потомков.

Каждая вершина бинарного дерева является структурой, состоящей из

четырех видов полей. Содержимое этих полей соответственно:

• информационное поле (ключ вершины);

• служебное поле (их может быть несколько или ни одного);

• указатель на левое поддерево;

• указатель на правое поддерево.

Виды бинарных деревьев:

* Строгие
* Нестрогие
* Полное
* Неполное

1. **Назовите основные операции над бинарными деревьями.**

Основные операции, осуществляемые с бинарными деревьями:

• создание бинарного дерева;

• печать бинарного дерева;

• обход бинарного дерева;

• вставка элемента в бинарное дерево;

• удаление элемента из бинарного дерева;

• проверка пустоты бинарного дерева;

• удаление бинарного дерева.

1. **Опишите поиск на основе деревьев.**

Поиск – это процесс нахождения конкретной информации в ранее созданном

множестве данных.

Для поиска данных необходимо использование соответствующих алгоритмов в

зависимости от следующих факторов:

• способ представления данных;

• упорядоченность множества поиска;

• объем данных;

• расположение данных во внешней или во внутренней памяти.

Анализируя дерево только с точки зрения представления данных в виде

иерархической структуры, заметим, что выигрыша при организации поиска не

получится. Сравнение ключа поиска с эталоном необходимо провести для всех

элементов дерева.

Уменьшить число сравнений ключей с эталоном возможно, если выполнить

организацию дерева особым образом, то есть расположить его элементы

по определенным правилам. При этом в процессе поиска будет просмотрено не

все дерево, а отдельное поддерево. Такой подход позволяет классифицировать

деревья в зависимости от правил построения.

1. **Что такое двоичное упорядоченное дерево?**

Двоичное дерево упорядоченно, если для любой его вершины x справедливы

следующие свойства:

• все элементы в левом поддереве меньше элемента, хранимого в x;

• все элементы в правом поддереве больше элемента, хранимого в x;

• все элементы дерева различны.

Если в дереве выполняются первые два свойства, но встречаются одинаковые

элементы, то такое дерево является частично упорядоченным.

Основные операции, производимые с упорядоченным деревом:

• поиск вершины;

• добавление вершины;

• удаление вершины;

• вывод (печать) дерева;

• очистка дерева.

1. **Как формируется случайное дерево?**

Случайные деревья поиска представляют собой упорядоченные бинарные

деревья поиска, при создании которых элементы (их ключи) вставляются в

случайном порядке.

При создании таких деревьев используется тот же алгоритм, что и при

добавлении вершины в бинарное дерево поиска.

1. **В чем заключается преимущество оптимального дерева?**

В двоичном дереве поиск одних элементов может происходить чаще, чем

других, то есть существуют вероятности k p поиска k-го элемента и для

различных элементов они неодинаковы. Можно предположить, что поиск в

дереве в среднем будет более быстрым, если те элементы, которые ищут чаще,

будут находиться ближе к корню дерева.

Дерево поиска с минимальной ценой называется оптимальным. То есть оптимальное бинарное дерево поиска – это бинарное дерево поиска, построенное в расчете на обеспечение максимальной производительности при заданном

распределении вероятностей поиска требуемых данных.

1. **Что такое АВЛ-дерево?**

АВЛ-дерево – структура данных, изобретенная в 1968 г. двумя советскими

математиками: Георгием Максимовичем Адельсон-Вельским и

Евгением Михайловичем Ландисом – и названная по первым буквам их

фамилий. Прежде чем дать конструктивное определение АВЛ-дереву, сделаем

это для сбалансированного двоичного дерева поиска.

Сбалансированным называется такое двоичное дерево поиска, в котором высота

каждого из поддеревьев, имеющих общий корень, отличается не более чем на

некоторую константу k, и при этом выполняются условия, характерные для

двоичного дерева поиска.

Идеально сбалансированным называется дерево, у которого для каждой вершины выполняется требование: число вершин в левом и правом поддеревьях

различается не более чем на 1.

Двоичное дерево считается сбалансированным по АВЛ, если для каждой вершины выполняется требование: высота левого и правого поддеревьев различаются не более чем на 1 (константа k = 1).

Не всякое сбалансированное по АВЛ дерево идеально сбалансировано, но

всякое идеально сбалансированное дерево сбалансировано по АВЛ.

Высотой узла (поддерева) называется максимальная длина пути, ведущего от

этого узла к самому нижнему узлу, являющемуся листом (иными словами

количество узлов в самой длинной ветви в этом поддереве).

1. **Расскажите о коэффициенте сбалансированности узла АВЛ-дерева.**

Корневой узел имеет высоту, равную высоте всего дерева.

АВЛ-дерево является частным случаем двоичного дерева поиска с k = 1.

Для узлов АВЛ-дерева определено понятие коэффициента сбалансированности

(balance factor).

Коэффициент сбалансированности (balance factor) – это разность высот правого и

левого поддеревьев, принимающая одно значение из множества {–1, 0, 1}.

При рассмотрении АВЛ-дерева следует учитывать, что если коэффициент

сбалансированности отрицателен, то узел ≪перевешивает влево≫, так как

высота левого поддерева больше, чем высота правого поддерева.

Если коэффициент сбалансированности больше нуля, то узел ≪перевешивает

вправо≫. Сбалансированный по высоте узел имеет коэффициент

сбалансированности, равный 0, что следует из того, что в АВЛ-дереве показатель сбалансированности должен быть в диапазоне [–1, 1].

1. **Дайте определение малого и большого вращения АВЛ-дерева. Для че го они применяются?**

Если после выполнения операции добавления или удаления коэффици-

ент сбалансированности какого-либо узла АВЛ-дерева становиться равен 2

(разница высот поддеревьев превышает 1, что приводит к разбалансировке),

т. е. | h(Ti ,R) − h(Ti , L) |= 2 , то необходимо выполнить операцию балансировки.

Она осуществляется путем вращения (поворота) узлов – изменения связей в

поддереве. Вращения не изменяют свойств бинарного дерева поиска и выпол-

няются за константное время. Всего различают четыре их типа:

1) малое правое вращение;

2) большое правое вращение;

3) малое левое вращение;

4) большое левое вращение.

Оба типа больших вращений являются комбинацией малых вращений

(право-левым или лево-правым вращением).

**6 Графы**

1. **Что такое граф?**

Граф – это совокупность двух конечных множеств: множества точек и множества линий, попарно соединяющих некоторые из этих точек. Множество точек

называются вершинами (узлами) графа. Множество линий, соединяющих

вершины графа, называются ребрами (дугами) графа

1. **Расскажите об ориентированных и неориентированных графах.**

Ориентированным графом (сокр. орграф) называется граф, состоящий из вершин, соединенных ребрами, имеющими направление.

Для неориентированного же графа, в отличие от ориентированного,

характерно следующее: каждое ребро неориентированного графа G = (V, E)

представляет собой неупорядоченную, т. е. ненаправленную пару вершин.

1. **Какие Вы знаете способы представления графов в ЭВМ?**

Классическим способом представления графа принято считать представление

с помощью матрицы инциденций.

**Матрица инциденций** – это двумерный массив A размерности n x m , в котором

указываются связи между инцидентными элементами графа (ребро и вершина). Столбцы матрицы соответствуют ребрам, строки – вершинам.

Более эффективным способом представления графов является представление

с помощью **матрицы смежности.**

Гораздо экономичнее с точки зрения памяти является представление графа с

помощью **списка пар**, соответствующих его ребрам.

Наилучшим способом представления графа в большинстве

случаев является структура данных, называемая **списками смежности**.

1. **Что такое матрица смежности и матрица инциденций?**

**Матрица инциденций** – это двумерный массив A размерности n x m , в котором

указываются связи между инцидентными элементами графа (ребро и вершина). Столбцы матрицы соответствуют ребрам, строки – вершинам.

Более эффективным способом представления графов является представление

с помощью **матрицы смежности.**

1. **Что такое список смежности и список ребер?**

Гораздо экономичнее с точки зрения памяти является представление графа с

помощью **списка пар**, соответствующих его ребрам.

Наилучшим способом представления графа в большинстве

случаев является структура данных, называемая **списками смежности**.

1. **Опишите алгоритм поиска в глубину в графе.**

При поиске в глубину посещается первая вершина, затем необходимо идти

вдоль ребер графа до попадания в тупик. Вершина графа является тупиком,

если все смежные с ней вершины уже посещены. После попадания в тупик

нужно возвращаться назад вдоль пройденного пути, пока не будет обнаружена

вершина, у которой есть еще непосещенная смежная вершина, а затем необхо-

димо двигаться в этом новом направлении. Процесс оказывается завершенным

при возвращении в начальную вершину, причем все смежные с ней вершины

уже должны быть посещены.

1. **Опишите алгоритм поиска в ширину в графе.**

При поиске в ширину после посещения первой вершины посещаются все

соседние с ней вершины. Потом посещаются все вершины, находящиеся на

расстоянии двух ребер от начальной. При каждом новом шаге посещаются

вершины, расстояние от которых до начальной на единицу больше

предыдущего. Чтобы предотвратить повторное посещение вершин, необходимо вести список посещенных вершин. Для хранения временных данных,

необходимых для работы алгоритма, используется очередь – упорядоченная

последовательность элементов, в которой новые элементы добавляются в

конец, а старые удаляются из начала.

1. **Что такое остовное дерево графа?**

Остовным деревом графа G называется свободное дерево, содержащее все

вершины V графа G.

1. **Приведите алгоритм Прима.**

Одним из методов построения остовного дерева минимальной стоимости

для помеченного графа G = (V,E) является алгоритм Прима. В ходе данного

алгоритма строится множество вершин U, из которого ≪вырастает≫ остовное

дерево. Пусть V = {1, 2,..., n} . Положим сначала U ={1}. На каждом шаге данного

алгоритма находится ребро наименьшей стоимости (u, v), такое, что u∈U и

v∈V \U , затем вершина v переносится из множества V \U во множество U.

Этот процесс продолжается до тех пор, пока множество U не станет равным

множеству V

1. **Приведите алгоритм Крускала.**

Алгоритм Крускала объединяет вершины графа в несколько связных компонент, каждая из которых является деревом. На каждом шаге из всех ребер,

соединяющих вершины из различных компонент связности, выбирается ребро с наименьшим весом. Необходимо проверить, является ли оно безопасным.

Ребро, являющееся самым легким из всех ребер, выходящих из данной

компоненты, будет безопасным.

Остается понять, как реализовать выбор безопасного ребра на каждом шаге.

Для этого в алгоритме Крускала все ребра графа G перебираются по

возрастанию веса. Для очередного ребра проверяется, не лежат ли концы ребра в разных компонентах связности, и если это так, ребро добавляется и

компоненты объединяются.

1. **Расскажите об эйлеровых циклах в графе.**

Циклом в неориентированном графе называется путь, у которого совпадают

начальная и конечная вершины. Цикл называется простым, если в нем нет

одинаковых вершин (кроме первой и последней).

Если граф имеет цикл (не обязательно простой), содержащий все ребра

графа по одному разу, то такой цикл называется эйлеровым циклом, а граф

называется эйлеровым графом. Если граф имеет путь (не обязательно простой),

содержащий все ребра по одному разу, то такой путь называется эйлеровым

путем, а граф называется полуэйлеровым графом.

Эйлеров цикл содержит не только все ребра (по одному разу), но и все

вершины графа (возможно по нескольку раз). Ясно, что эйлеровым может быть

только связный граф. Очевидно, что не все даже связные графы эйлеровы.

1. **Что Вы знаете о гамильтоновых циклах в графе?**

Гамильтоновым путем графа называется его простой путь, который проходит

через каждую вершину графа точно один раз.