1 Билет. Понятие системы счисления, виды СС, плюсы и минусы каждого вида.

Система счисления — это совокупность правил записи чисел посредством конечного набора символов.

Виды СС:
1) Позиционные системы счисления:
В позиционных системах счисления один и тот же числовой знак в записи числа имеет различные значения в зависимости от того места, где он расположен. Привычная нам СС, легко производить вычисления.

2) Смешанные системы счисления:
Называется система счисления, в которой числа, заданные в некоторой системе счисления с основанием P изображаются с помощью цифр другой системы счисления с основанием Q, где Q<P. В такой системе **P называется старшим основанием**, Q — **младшим основанием**, а сама система счисления называется **Q-P-ричной.**

3) Непозиционные системы счисления:
В непозиционных системах счисления величина, которую обозначает цифра, не зависит от положения в числе. При этом система может накладывать ограничения на положение цифр, например, чтобы они были расположены в порядке убывания. Устаревший вид СС, над ним трудно производить математические операции.

2 Билет. Распространённые СС с примерами

1) Римская система счисления является непозиционной системой.(Тяжело производить вычисления, длинная запись числа, много символов для запоминания)

2) Десятичная система счисления. (Привычная нам СС, легко производить вычисления, всего 10 знаков)

3) Двоичная система счисления (Самая экономичная СС, используется в ЭВМ, не легка для вычислений)

4)Восьмеричная система счисления (Компактнее двоичной)

5) Шестнадцатеричная система счисления (Компактнее восьмеричной)

6) Вавилонская система счисления является позиционной системой. (Громоздкая запись чисел, неудобна для счета, неактуальность)

7) Троичная система счисления - Азбука Морзе

3 Билет. Экономичность СС с подробным объяснением

Из нескольких СС наиболее экономичной будет та, которая позволяет с помощью определенного количества заданных базисных знаков записать наибольшее количество чисел.

Пусть задано 60 базисных знаков.

2 знака
2СС ⇒ 30 групп
2^30 чисел

3 знака
3 СС ⇒ 20 групп
3^20 чисел
12 знаков
12 СС ⇒ 5 групп
12^5 чисел

Условием сравнения пол экономичности является заданная в виде константы количество базисных знаков.

4 Билет. Сложности понятия и формы восприятия информации.

Свойства:

1) Новизна (2*2=4) – новые сведения.

2) Понятность (Производная в 3 классе).

3) Объем информации:

А) Неожиданность получения инф. сообщения (чем больше неожиданность, тем больше инф. получено, например, нобелевская премия у ученика школы и ученого).

Б) Заинтересованность (чем больше заинтересованность, тем больше объем информации, например, студент, бизнесмен, доллар)

В) Вероятность получения инф. сообщения (чем больше вероятность, тем меньше объем инф., например, Пермь, Токио, Цунами)

Г) Последействие, получившего инф. сообщение. (чем активнее последействие, тем больший объем получен)

Формы восприятия информации:

1) Знаковая: текст, пиктограммы, ноты, цифры, иероглифы.

2) Образная форма - связанная со сравнением.

5 Билет. Формы обработки сообщений , понятие информации.

Формы обработки сообщений:

1) С помощью вычислений – формулы, нотная грамота.

2) Логические умозаключения – взаимосвязь по принципу ЕСЛИ, ТО, расследования, банковские и биржевые кодировки, курсы валют.

3) С помощью сортировки – упорядочивание данных по некоторому принципу.

4) Обработка изменяющая форму сообщения, но не изменяющая содержания – перевод языков, кодирование инф, перевод СС, конспект лекций.

5) Поисковая форма – построение критериев запроса.

Информация – сообщение, уменьшающее неопределенность нашего знания об исследуемом объекте. (в 2 раза 1 бит)

6. Этапы развития человеческой цивилизации

Период	Наука	Технология	Ресурсы
Освоение вещества(материи)	Освоение материи и осознание единства материального мира	Огонь, колесо, крыло, орудие материального производства, разделение труда	Полезные ископаемые (металл и т.д.)
Освоение энергии	Единство мира материи + Энергия	Накопление энергии (аккумуляторы) Передача (сети) Преобразование (трансформаторы)	Вода, солнце, ветер, топливо, атомная энергия
Освоение информации	Материя + Энергия + Информация мира	Накопление информации (бумага, магнитные и лазерные и т.д.) Передача информации (СМИ, интернет, художественные объекты, музыка, речь, письмо, соц. сети, органы чувств. Обработка информации (перевод из одного вида в другой, формулы, шифр, кодирование, разработка. Средства обработки – от линейки до компьютера.	Передача знаний (литература, историческое наследие, от человека к человеку, алфавит, квалификация, патенты и авторское дело.

7 Билет. Этапы развития ИТ и КТ.

СМ В БЕСЕДЕ ТАБЛИЦУ EXCEL

8 Билет. Интеллектуальные программные системы из 4 этапа развития ИТ.

Интеллектуальные системы - это технические или программные системы, которые реализуют некоторые черты человеческого интеллекта, дающие возможность осиливать трудные задачи, решение которых человеком в реальное время невозможно.

1) Система ситуационного моделирования (реакция на ситуацию: если – то).

2) Системы имитационного моделирования (стоится по уравнениям) - метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему (построенная модель описывает процессы так, как они проходили бы в действительности).

3) Системы когнитивного моделирования – это системы, которые позволяют построить прогноз (на основе статистики, сбора и анализа информации).

9 Билет. Современные устройства ввода и вывода.

1) Сканер (большая логистическая система).

2) Чувствительный экран.

3) Трекбол (мышь наоборот).

4) Плата спутникового приёма (работа ГИБДД, управление пожарной ситуацией, GPS, безопасный город).

5) Большая мышь (управление КТ ногами).

6) Модем - устройство преобразования сигнала (высокочастотные ßà низкочастотные) (Пример: телефония, связано со скоростью передачи звуковых сигналов).

7) Виртуальные шлем и перчатки (технологии тренажа: врач проводит операцию, обучение вождению, управление башенным краном).

8) Устройство для снятия гальванического потенциала кожи - РН (использует скорая помощь).

9) Сканирование движение глазных яблок (скорая помощь, уменьшение смертности при ДТП).

10) 3D-принтер.

11) Web-камера.

12) Трекпоинт или тензометрический джойстик (миниатюрный джойстик (замена мыши)).

13) Графический планшет и световое перо.

14) Устройства видео-захвата (камеры видеонаблюдения).

15) TV-тюнер (телевизор).

10 Билет. Характеристики современных компьютерных и информационных технологий.

1) Любые технологии должны быть определены в общепринятых терминах (заказчик всегда знает ЧТО, а разработчик всегда знает КАК). Эти два направления: ЧТО и КАК должны быть реализованы в одних и тех же терминах.

2) Проклятие размерности связано с тем, что по ходу разработки проекта заказчик загружает модель дополнительными всё более сложными функциями (разработчик должен выделить главное и отсечь то, чем можно пренебречь).

3) Любой проект проходит стадию моделирования (обычно когнитивного моделирования), которая идёт в специальных программных средах.

Достоинства:

1. Участие, как заказчика, так и разработчика.

2. Модель позволяет провести оперативное изменение структуры, изменить связь между блоками проекта, изменить коэффициенты и параметры.

4) Модель должна быть визуализирована, наглядно представлена в виде структур, диаграмм, интерактивных рисунков, так как результат виден там, где получено хорошее визуальное отображение.

5) Человеческий интеллект спонтанно реагирует в острых ситуациях (немонотонность человеческой логики).

А информационная система в острых (нестандартных) ситуациях вынуждена принимать решение, заложенное в её алгоритме.

6) Границы задачи/проекта должны быть заранее определены, чтобы избежать его расширения.

11 Билет. Измерение информации понятие бита формула Хартли для цифровой информации формула Шеннона.

Бит[bit – binary digit] – минимальная единица измерения информации.

Бит – минимальный объём памяти компьютера.

Бит соответствует одному двоичному разряду и может принимать значения 1 или 0.

Уменьшение неопределённости знания об исследуемом объекте в 2 раза несёт 1 бит информации.

Вопрос: «Как измерить информацию?» очень непростой. Ответ на него зависит от того, что понимать под информацией. Но поскольку определять информацию можно по-разному, то и способы измерения тоже могут быть разными:

1) Содержательный. Сообщение – информационный поток, который несёт информацию для человека, если содержащиеся в нём сведения являются для него новыми и понятными. Сообщение должно быть информативно, в противном случае количество информации, с точки зрения человека, равно 0. Пример: вузовский учебник по высшей математике содержит знания, но они не доступны первокласснику.

2) Алфавитный. Удобен при использовании технических средств работы с информацией, так как НЕ зависит от содержания сообщения. Количество информации зависит от объёма текста и мощности алфавита. Если события равновероятны, то количество информации можно рассчитать по формуле Хартли:

3) Вероятностный. Все события происходят с различной вероятностью, но зависимость между вероятностью событий и количеством информации, полученной при совершении того или иного события, можно выразить формулой, которую предложил Шеннон (для НЕравновероятных событий):

Формула Шеннона

$$I = -\sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}$$

где
I – количество информации (бит);
N – количество возможных событий;
p_i –вероятность i-го события.

12 Билет. Формула Хартли для символьной информации алфавитный подход и для неравновероятных событий формула Шенона.

Алфавитный подход удобен при использовании технических средств работы с информацией, так как НЕ зависит от содержания сообщения. Количество информации зависит от объёма текста и мощности алфавита. Если события равновероятны, то количество информации можно рассчитать по формуле Хартли.

Формула Хартли позволяет вычислить количество полученной информации в битах через вычисления меры уменьшения неопределённости нашего знания при получении информационного сообщения.

$$N=2^{i}$$

Где:

N – количество возможных РАВНОВЕРОЯТНЫХ исходов события.

i – количество информации в битах, которая получена в результате сообщения, ликвидирующее неопределённость.

Вероятностный подход. Все события происходят с различной вероятностью, но зависимость между вероятностью событий и количеством информации, полученной при совершении того или иного события, можно выразить формулой, которую предложил Шеннон (для НЕравновероятных событий):

13 Билет. Алгоритм Маркова.

• Нормальные алгоритмы Маркова – аппарат преобразования слов по некоторым правилам.

• Слово – любая конечная последовательность символов, входящих в алфавит.

• Алфавитом называется любое непустое множество. Его элементы называются буквами, а любые последовательности букв — словами в данном алфавите.

• Нормальные алгоритмы Маркова - совокупность правил подстановки S1, S1, …, Sn.

S1: U1 -> V1

S2: U2 -> V2

…

Sn: Un -> Vn

//Преобразование слова состоит в поиске некоторого подслова, совпадающего с левой частью какого-либо правила подстановки, и замене этого подслова правой частью данного правила подстановки.

Завершение преобразования слова

1) На некотором шаге преобразования ни одно из правил оказывается неприменимым к слову, т.е. ни одна из левых частей в слове не содержится.

2) Правила подстановки бывают двух видов: НЕТЕРМИНАЛЬНЫЕ и ТЕРМИНАЛЬНЫЕ. Если на некотором шаге преобразования было применено терминальное правило, то процесс преобразований считается завершенным, т.е. новый просмотр правил не производится

Обозначения: Терминальные (-->) / Нетерминальные (|-->)

Особенности работы нормальных алгоритмов Маркова:

1) Если левая часть некоторого правила входит в слово более одного раза, то заменяется только самое левое вхождение (на одном шаге преобразования).

2) После каждого применения к слову некоторого правила Si новый просмотр правил всегда начинается с правила S1.

4) Левая часть каждого правила может быть пустой. Действие такого правила состоит в том, что к началу слова добавляется правая часть правила. Правило с пустой частью считается применимым к любому слову, это означает, что оно должно всегда быть последним в списке правил, так как оно блокирует все последующие правила. //Для завершения работы такого алгоритма в нем должно присутствовать хотя бы одно терминальное правило подстановки.

5) Правая часть правила также может быть пустой. Результатом применения такого правила является удаление из слова последовательности символов, совпадающих с левой частью правила.

Примеры нормальных Алгоритмов Маркова

Пример 1. A={a,b}. Преобразовать слово P так, чтобы в его начале оказались все символы a, а в конце – все символы b. Например: babba ⇒ aabbb.

$$\begin{cases} ba \to ab \end{cases}$$

Пример 2. A={a,b}. Требуется приписать символ a к концу слова P.

Например: bbab → bbaba

$$\begin{cases} *a \to a* \\ *b \to b* \\ * \mapsto a \\ \quad \to * \end{cases}$$

14 Билет. Машина Тьюринга.

Абстрактная машина Тьюринга – это простая и полезная абстрактная модель вычислений (компьютерных и цифровых), которая является достаточно общей для воплощения компьютерной задачи.

Состав МТ:

1. Бесконечная в обе стороны лента, которая состоит из ячеек.

В ячейках расположены базисные символы, входящие во множество A (алфавит).

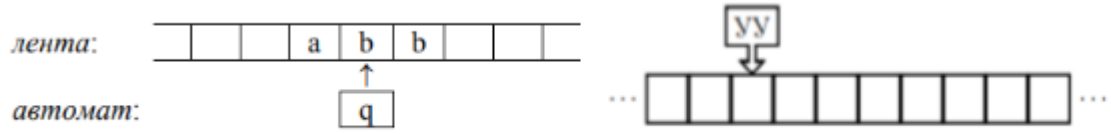
$$A = \{a_0, a_1, \ldots, a_m\}$$

2. Голова машины Тьюринга (автомат) – всегда указывает на какую-то ячейку, за счет перемещения по ленте позволяет считывать, стирать и записывать содержание текущей ячейки.

Состояние головы определяется множеством Q, которое состоит из управляющих команд Q1, Q2, …, Qn.

Команда, которую выполняет голова, является действием, которое задает программист.

3. Устройство управления головой - передает команды из множества Q на голову.



Конфигурацией МТ называется таблица, которая связывает данные на ленте с выполнением заданных команд в виде таблицы алгоритмов.

Такт работы машины Тьюринга:

МТ работает тактами (по шагам), которые выполняются один за другим. На каждом такте автомат МТ выполняет следующие действия:

1) записывает некоторый символ S¢ в ячейку.

// пустое содержимое ячейки принято называть символом «пусто» и обозначать знаком Λ («лямбда»).

2) сдвигается на одну клетку влево (обозначение – L), вправо (R), либо остается неподвижным (N).

3) переходит в некоторое состояние q¢ (в частности, может остаться в прежнем состоянии). Действия одного такта записываются в виде:

Q, S à Q’, S‘, [L, R, N], (или S’, [L, R, N], Q'),

где конструкция с квадратными скобками означает возможность записи в этом месте любой из букв L, R или N. Например, такт

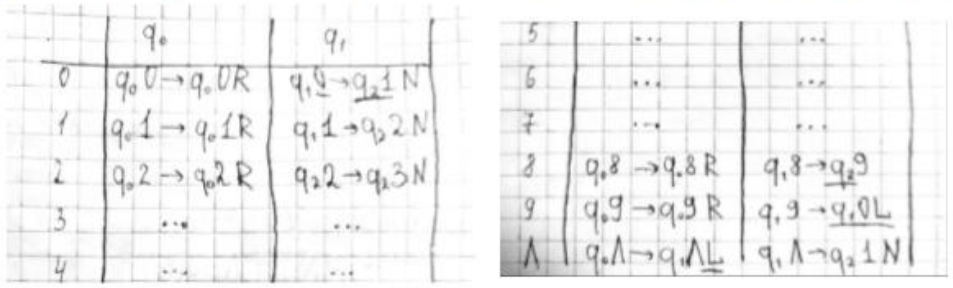
(или a,R,q2) означает запись символа a в видимую клетку на место символа 1, сдвиг на одну клетку вправо и переход в состояние q2.

Примеры программы для машины Тьюринга

Пример 1. A = {0,1,2,3,4,5,6,7,8,9}. Пусть P – непустое слово; значит, P – это последовательность из десятичных цифр, т.е. запись неотрицательного целого числа в десятичной системе. Требуется получить на ленте запись числа, которое на 1 больше числа P.

В виде программы для МТ эти действия описываются следующим образом:

В виде программы для МТ эти действия описываются следующим образом:

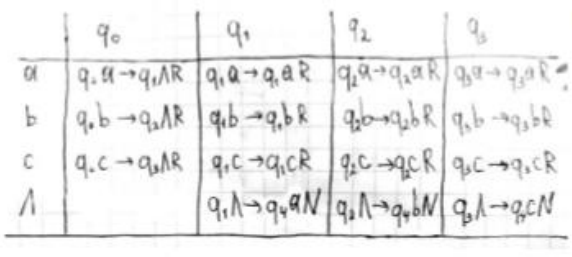


Или

	0	1	2	3	4	5	6
q1	,R,	,R,	,R,	,R,	,R,	,R,	
q2	1, ,!	2, ,!	3, ,!	4, ,!	5, ,!	6, ,!	

	6	7	8	9	Λ	*комментарий*
q1	,R,	,R,	,R,	,R,	,L,q2	под последнюю цифру
q2	7, ,!	8, ,!	9, ,!	0,L,	1, ,!	видимая цифра + 1

Пример 2. A={a,b,c}. Перенести первый символ непустого слова P в его конец.



или

	a	b	c	Λ
q1	Λ,R,q2	Λ,R,q3	Λ,R,q4	,R,
q2	,R,	,R,	,R,	a, ,!
q3	,R,	,R,	,R,	b, ,!
q4	,R,	,R,	,R,	c, ,!

15 Билет. Перевод целого десятичного числа в компьютерную форму.

Диапазон целых чисел, которые могут быть представлены в памяти ЭВМ зависит от ячеек памяти, используемых для хранения таких чисел

В k разрядной ячейке может храниться 2^k различных чисел

Беззнаковые числа

n-количество битов в машинном слове (МС) минимальное значение, которое может принимать целое беззнаковое число 0, максимальное 2^k-1

Перевод:

1) Перевод в 2СС

2) Запись в машинное слово с k разрядами справа

3) Заполнение нулями недостающие разряды слева

4) Конвертируем двоичное число в 16-тиричное

Числа со знаком

Самый старший разряд ячейки памяти(слева) отводится под знак числа, а остальные под само число. 0 используется для +, 1 для –

Для работы с отрицательными числами

1) Перевод модуля числа в 2СС

2) Инвертируем его (1 меняем на 0, 0 на 1)

3) Прибавляем 1

Пример:

МС=8 бит, Число -25, его модуль в 2СС = 11001

1)записываем в ячейки памяти модуль

0	0	0	1	1	0	0	1

2) Инвертируем

1	1	1	0	0	1	1	0

3)Прибавляем 1

1	1	1	0	0	1	1	1

16 Билет. Перевод действительного десятичного числа в компьютерную форму.

±	Порядок							Мантиса числа																			

1. Перевести модуль в 2 СС с 24-мя значащими знаками

2. Привести к нормализованной форме по формуле R=m*np(R-число, m-мантиса, p-математический порядок)

3. Найти машинный порядок по формуле Mp=p+64(в 10СС) или Mp=p+1000000(в 2СС)

4. Вписываем знак, машинный порядок и мантиссу в МС(1 для -, 0 для +)

5. Переводим МС в 16 СС.

17 Билет. Цели кодирования.

Кодирование – это формирование представления информации в виде перехода от формы, понятной человеку, к форме, удобной для хранения, передачи и обработки.

Цели кодирования:

1) Удобство физической реализации (связано техникой);

2) Удобство восприятия информации (на той стороне, где находится итоговый код);

3) Высокая скорость передачи и обработки кода;

4) Экономичность (уменьшение избыточности информации сообщения – синонимы);

5) Надежность – это защита от случайных искажений;

6) Сохранность информативности сообщений, защита от нежеланного доступа.

На разных этапах использования информации сообщения неоднократно перекодируются, поэтому в качестве цели не может быть применен один критерий, он всегда комплексный. Например, цель экономичности может уменьшать надежность и удобство восприятия.

18 Билет. Понятие математического и машинного порядков.

± машинный порядок	М А	Н Т И С	С А

В старшем бите 1-го байта хранится знак числа. В этом разряде 0 обозначает плюс, 1 — минус. Оставшиеся 7 бит первого байта содержат машинный порядок. В следующих трех байтах хранятся значащие цифры мантиссы.

Что такое машинный порядок? В семи двоичных разрядах помещаются двоичные числа в диапазоне от 0000000 до 1111111. В десятичной системе это соответствует диапазону от 0 до 127. Всего 128 значений. Знак порядка в ячейке не хранится. Но порядок, очевидно, может быть как положительным так и отрицательным. Разумно эти 128 значений разделить поровну между положительными и отрицательными значениями порядка. В таком случае между машинным порядком и истинным (назовем его математическим) устанавливается следующее соответствие:

Машинный порядок	0	1	2	3	...	64	65	...	125	126	127
Математический порядок	-64	-63	-62	-61	...	0	1	...	61	62	63

Если обозначить машинный порядок Мр, а математический — р, то связь между ними выразится такой формулой:

Мр = р + 64.

Итак, машинный порядок смещен относительно математического на 64 единицы и имеет только положительные значения.

19 Билет. Логические операции.

1) Логическое умножение или конъюнкция:

Конъюнкция - это сложное логическое выражение, которое считается истинным в том и только том случае, когда оба простых выражения являются истинными, во всех остальных случаях данное сложенное выражение ложно.

Обозначение: F = A ^ B.

Таблица истинности для конъюнкции

A B F

1 1 1

1 0 0

0 1 0

0 0 0

2) Логическое сложение или дизъюнкция:

Дизъюнкция - это сложное логическое выражение, которое истинно, если хотя бы одно из простых логических выражений истинно и ложно тогда и только тогда, когда оба простых логических выраженья ложны.
Обозначение: F = A v B.

Таблица истинности для дизъюнкции

A B F

1 1 1

1 0 1

0 1 1

0 0 0

3) Логическое отрицание или инверсия:

Инверсия - это сложное логическое выражение, если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное логическое выражение ложно, то результат отрицания будет истинным. Другими простыми слова, данная операция означает, что к исходному логическому выражению добавляется частица НЕ или слова НЕВЕРНО, ЧТО.

Обозначение: F = ¬A.

Таблица истинности для инверсии

A ¬A

1 0

0 1

4) Логическое следование или импликация:

Импликация - это сложное логическое выражение, которое истинно во всех случаях, кроме как из истины следует ложь. То есть данная логическая операция связывает два простых логических выражения, из которых первое является условием (A), а второе (B) является следствием.

«A → B» истинно, если из A может следовать B.

Обозначение: F = A → B.

Таблица истинности для импликации

A B F

1 1 1

1 0 0

0 1 1

0 0 1

5) Логическая равнозначность или эквивалентность:

Эквивалентность - это сложное логическое выражение, которое является истинным тогда и только тогда, когда оба простых логических выражения имеют одинаковую истинность.

«A ↔ B» истинно тогда и только тогда, когда A и B равны. Обозначение: F = A ↔ B. Таблица истинности для эквивалентности

A B F

1 1 1

1 0 0

0 1 0

0 0 1

6) Операция XOR (исключающие или)

«A ⊕ B» истинно тогда, когда истинно A или B, но не оба одновременно. Эту операцию также называют "сложение по модулю два".

Обозначение: F = A ⊕ B.

A B F

1 1 0

1 0 1

0 1 1

0 0 0

20 Билет. Виды информации.

1)Структурная информация

Это такая информация, которой обладают все материальные объекты. Является устойчивой и свойством материи. Существуют в природе в потенциальной форме. При накоплении объектом структурной информации уровень организации этого объекта повышается. Если человек воздействует на объект, то он изменяет структурную организацию и структурную информацию этого объекта => таким образом, структурная информация описывает устройство материи.

2)Оперативная информация

Используется в целях познания, управления, поглощения, преобразования информации => таким образом, оперативная информация содержится в процессе, в итоговой информации, и описывает состояние материи.

Дорожный знак содержит оба типа информации.

Актуализирует для каждого индивида морально—этические, правовые, идеологические и прочие нормы сегодняшнего общества, буквально—общества сегодняшнего дня.

Преимущественный канал распространения—СМИ (массовые коммуникации)

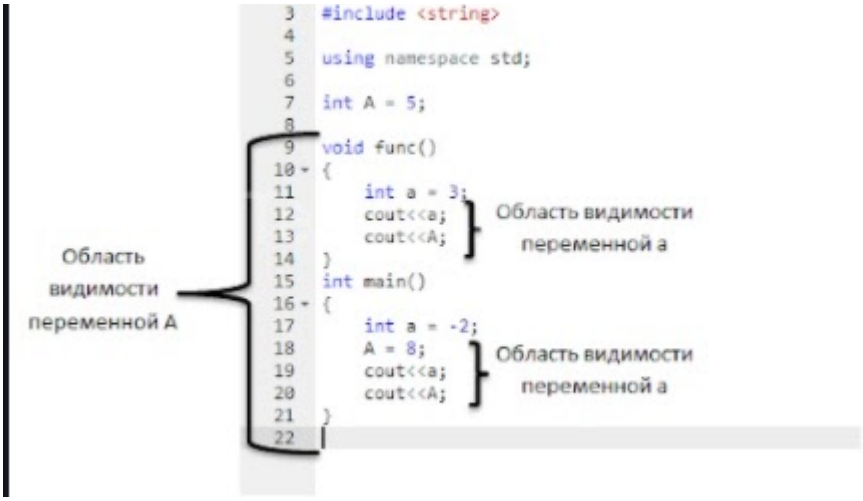
Доля оперативной информации возрастает в сознании человека массового общества.

3)Фундаментальная информация

Создает устойчивую ориентацию человека в обществе. Приобщает человека к профессии, науке, искусству. Формирует знания и интересы индивида.

21 Билет. Локальная и глобальная переменные. Описание переменных и классы памяти с примерами.

Каждая переменная имеет свою область видимости, то есть такую область, в которой можно работать с переменной. За пределами этой области, о данной переменной ничего известно не будет, а значит и использовать её нельзя. Итак, переменная находится в области видимости, если к ней можно получить доступ.



Существуют локальные и глобальные переменные. Так вот, переменные, объявленные внутри функции, называются локальными.

Разделение переменных на глобальные и локальные соответствует одному из главных правил программирования, а именно – принципу наименьших привилегий. То есть, переменные, объявленные внутри одной функции, должны быть доступны только для этой функции и ничему другому, в конце концов, они создавались именно для этой функции.

Локальные переменные имеют свои области видимости, этими областями являются функции, в которых объявлены переменные. Таким образом, в разных функциях можно использовать переменные с одинаковыми именами, что в свою очередь очень удобно.

Глобальные переменные объявляются вне тела какой-либо функции, и поэтому область видимости таких переменных распространяется на всю программу

22 Билет. Оператор switch и тип данных enum.

Использование инструкции switch

Инструкция switch используется для создания в коде множества точек ветвления. Вот упрощенное представление инструкции switch:

```
switch (choice)
{
    case value1:
        statement1;
```



```
        break;
    case value2:
        statement2;
        break;
    case value3:
        statement3;
        break;
    case valueN:
statementN;
break;
    default:
statementN + 1;
}
```

Данная инструкция по порядку сравнивает варианты (choice) с возможными значениями (value1, value2 и value3). Если вариант равен одному из значений, то программа выполняет инструкцию (statement), соответствующую этому значению. Встретив инструкцию break, программа покидает структуру switch. Если choice не совпадает ни с одним значением, то выполняется инструкция, ассоциированная с факультативным вариантом default.

Использовать break и default необязательно. Но если опустить break, то программа продолжит выполнять остальные инструкции, пока не встретит break или default либо пока не закончится инструкция switch. Как правило, инструкция break ставится в конце каждого условия case.

СОВЕТ

Хотя условие default не является обязательным, его удобно ставить в конце как универсальный уловитель (catchall).

Закрепим эту теорию на примере. Допустим, choice равно value2. Сначала программа сравнит choice с value1. Поскольку два этих значения не равны, выполнение программы продолжится. Далее программа сравнит choice с value2. Поскольку эти значения равны, программа выполнит statement2. Затем программа дойдет до инструкции break и выйдет из структуры switch.

ОСТОРОЖНО!

Инструкция switch подходит только для тестирования целочисленных значений int либо любого значения, которое можно интерпретировать как int — например, char или enumerator. Инструкция switch не сработает с данными других типов.

Перечисляемые типы

Перечисление (или ***«перечисляемый тип»***) — это тип данных, где любое значение (или ***«перечислитель»***) определяется как **символьная константа**. Объявить перечисление можно с помощью **ключевого слова enum**. Например:

```
// Объявляем новое перечисление Colors
enum Colors
{
    // Ниже находятся перечислители - все возможные значения этого типа данных
    // Каждый перечислитель отделяется запятой (НЕ точкой с запятой)
    COLOR_RED,
    COLOR_BROWN,
    COLOR_GRAY,
    COLOR_WHITE,
    COLOR_PINK,
    COLOR_ORANGE,
    COLOR_BLUE,
    COLOR_PURPLE, // о конечной запятой читайте ниже
}; // однако сам enum должен заканчиваться точкой с запятой

// Определяем несколько переменных перечисляемого типа Colors
Colors paint = COLOR_RED;
Colors house(COLOR_GRAY);
```

Объявление перечислений не требует выделения памяти. Только когда переменная перечисляемого типа определена (например, как переменная `paint` в примере, приведенном выше), только тогда выделяется память для этой переменной.

Обратите внимание, каждый перечислитель отделяется запятой, а само перечисление заканчивается точкой с запятой.

23 Билет. Решение нелинейных уравнений методом приближенного вычисления корней.

Дано:

· Функция f(x), непрерывная на интервале AB и пересекающая ось OX.

· Величина eps – допустимая точность вычисления корня, обычно eps = 10-6.

· Интервал AB, на котором функция непрерывна и имеет корень

Методы:

1) Метод половинного деления

1 шаг: на интервале AB берется точка X0 – середина AB.

2 шаг: необходимо выбрать отрезок Ax0 или Bx0, тот в котором f(x) пересекает OX.

Отбрасывается та половина, в которой корня точно нет.

Выбираем тот из отрезков, где произведения значений функции на его концах меньше нуля.

y(B) * y(X0) > 0 => интервал BX0 отбрасываем.

f(X0) * y(A) < 0 => интервал AX0 берем для дальнейшего рассмотрения.

3 шаг: переносим точку B в точку X0 и повторяем деления нового интервала AB пополам.

4 шаг: отбрасываем AX1 и точку A переносим в точку X1.

5 шаг: выполнение продолжается до тех пор, пока разность между 2 соседними корнями по абсолютной величине не станет меньше eps.

$$|X_{n-1}-X_n| <= eps$$

2) Метод Ньютона

1 шаг: выбирается начальное значение X0 – произвольная на интервале AB.

2 шаг: строится касательная к f(x) в точке X0; касательная пересекает ось OX в X1.

3 шаг: метод заканчивается с такой же проверкой.

$$|X_{n-1}-X_n| <= eps$$

$$X_n = X_{n-1} - F(X_{n-1})/F'(X_{n-1})$$

3) Метод итерации

1 шаг: X0 выбирается произвольная на интервале AB.

2 шаг: следующее значение находится как функция прошлого значения.

X1 = f(x0)

3 шаг: X2 = f(X1) => X3 = f(X2) и т.д.

4 шаг:

```
double IterationMethood
do
px = x;
double px, x = (left + right)/2; (double left, double right, doble eps)
x = 2 – sin(1/x);
while (abs(px – x) > eps);
return x;
```

24 Билет. Понятие указатели , их виды. Битовые поля и ссылки с примерами.

Указатель - это переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причем на самое его начало (Определение из гугла)

В C++ различают три вида указателей:

- указатели на объект
- указатели на функцию
- указатели на void

Ссылки C++:

```
int value = 7; // Инициализация переменной
int &link = value; // ссылка на переменную value

value = 8; // value теперь имеет значение 8
link = 9; // value теперь имеет значение 9
```

Примечание: чаще всего используется в функциях.

Битовые поля:

Битовые поля должны объявляться как int, unsigned, signed. Битовые поля длиной 1 должны объявляться как unsigned, поскольку 1 бит не может иметь знака.

ПРИМЕР

```
struct DataTime {
        unsigned short Day: 5;      // 5 бит (00000 - 11111) - максимальное число 31
        unsigned short Month: 4;  // 4 бит (0000 - 1111) - максимальное число 15
        unsigned short Year: 7;     // 7 бит (0000000 - 1111111) - максимальное число 127
} field;
Тип_данных : Количество_Бит_Для_Данных
```

Объявляем

```
DataTime d;
```

Одна из главных причин использования битовых полей - экономия.
В приведенном примере используется всего 16 бит = 2 байта. Для примера: int занимает 4 байта, а переменных нам понадобилось мы 3, т.е мы бы использовали 12 байт. Экономия в 6 раз

25 Билет. Способы инициализации указателей и операции с указателями. ПРИВЕДЕНИЕ ТИПОВ.

Способы инициализации указателя:

1. Присваивание указателю адреса существующего объекта:

a. с помощью операции получения адреса:

int a = 5; // целая переменная-

int* p = &a; //в указатель записывается адрес a

int* p (&a): // то же самое другим способом

b. с помощью значения другого инициализированного указателя:

int* r = p;

c. с помощью имени массива или функции, которые трактуются как адрес:

int b[10]; // массив

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

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

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

pf = f; // присваивание адреса функции

2. Присваивание указателю адреса области памяти в явном вид:

char* vp = (char *)0xB8000000;

Здесь OxB8000000 — шестнадцатеричная константа, (char *) — операция приведения типа: константа преобразуется к типу «указатель на char».

3. Присваивание пустого значения:

1nt* suxx = NULL;

1nt* rulez = 0;

В первой строке используется константа NULL, определенная в некоторых заголовочных файлах C как указатель, равный нулю. Рекомендуется использовать просто 0, так как это значение типа int будет правильно преобразовано стандартными способами в соответствии с контекстом.

Все, что ниже рассказывать не нужно, только если попросят!!!

4. Выделение участка динамической памяти и присваивание ее адреса указателю:

Указатели чаще всего используют при работе с динамической памятью («куча»).

Динамическая память – это свободная память, в которой можно во время выполнения программы выделять место в соответствии с потребностями.

Доступ к выделенным участкам динамической памяти, называемым динамическими переменными, производится только через указатели. Время жизни динамических переменных — от точки создания до конца программы или до явного освобождения памяти.

В C++ используется два способа работы с динамической памятью:

1) Через операции new и delete:

*int * n = new int: //1*

*int * m = new int (10); // 2*

*int * q = new int [10]; // 3*

1: операция new выполняет выделение достаточного для размещения величины типа int участка динамической памяти и записывает адрес начала этого участка в переменную n. Память под саму переменную n выделяется на этапе компиляции.

2: кроме описанных выше действий, производится инициализация

выделенной динамической памяти значением 10.

3: операция new выполняет выделение памяти под 10 величин типа

int (массива из 10 элементов) и записывает адрес начала этого участка в переменную q. которая может трактоваться как имя массива.

Если память выделить не удалось, по стандарту должно порождаться исключение bad_alloc. Старые версии компиляторов могут возвращать 0.

Освобождаем память с помощью delete:

delete n; delete m; delete [ ] q;

Если память выделялась с помощью new[], для освобождения памяти необходимо применять delete[]. Размерность массива при этом не указывается. Если квадратных скобок нет, то никакого сообщения об ошибке

не выдается, но помечен как свободный будет только первый элемент массива, а остальные окажутся недоступны для дальнейших операций. Такие ячейки памяти называются мусором.

2) Через семейство функций malloc, который достался в наследство от C:

int u = (int *)malloc(sizeof(int)); // 4*

4: операция malloc выполняет выделение достаточного для размещения величины типа int участка динамической памяти и записывает адрес начала этого участка в переменную u.

В функцию передается один параметр — количество выделяемой памяти в байтах. Конструкция (int) используется дляприведения типа указателя, возвращаемого функцией, к требуемому типу. Если память выделить не удалось, функция возвращает 0.*

Освобождаем память с помощью функции free:

free(u);

Операцию new использовать предпочтительнее, чем функцию malloc, особенно при работе с объектами.

Для того чтобы использовать malloc, требуется подключить к программе заголовочный файл <malloc.h>.

Операции с указателями

1) Разыменования (разадресации).

Предназначена для доступа к величине, адрес которой хранится в указателе. Эту операцию можно использовать как для получения, так и для изменения значения величины (если она не объявлена как константа).

2) Присваивания.

Заполняет область памяти некоторым значением.

Присваивание без явного приведения типов допускается:

а) Указателям типа void *;

b) Если тип указателей справа и слева от оператора присаивания один и тот же.

Пример:

char a; // переменная типа char

char * p = new char; /* выделение памяти под указатель и под динамическую переменную типа char */

*p = 'Ю'; a = *p; // присваивание значения обеим переменным

3) Получение адреса (&);

Применима к величинам, имеющим имя и размещенным в оперативной памяти. Таким образом, нельзя получить адрес скалярного выражения, неименованной константы или регистровой переменной.

Пример:

int t = 7, *p;

p = &t; //Присваиваем указателю p адрес переменной t

4) Арифметические операции:

- Автоматически учитывают размер типа величин, адресуемых указателями. Эти операции применимы только к указателям одного типа и имеют смысл в основном при работе со структурами данных, последовательно размещенными в памяти, например, с массивами.

a) Сложение с константой

Если указатель на определенный тип увеличивается или уменьшается на константу, его значение изменяется на величину этой константы, умноженную на размер объекта данного типа, например:

int a = 3, *p;

p = &a; // p = 012FFC04

p += 5; // p = 012FFC18 5*4(размер int) = 2010 = 1416

cout << “p = “<<*p; // Вывод: p = 1;

b) Инкремент - перемещает указатель к следующему элементу массива,

c) Декремент – перемещает указатель к предыдущему элементу массиву.

Фактически значение указателя изменяется на величину sizeof(тип).

Пример:

short * p = new short [5];

p++; // значение p увеличивается на 2

long * q = new long [5];

q++;// значение q увеличивается на 4

d) Разность двух указателей — это разность их значений, деленная на размер типа в байтах (в применении к массивам разность указателей, например: a[6] – a[3] = 3). Суммирование двух указателей не допускается.

5) Сравнение.

Пример:

int a = 4, b = 6, *p1, *p2;

p1 = &a;

p2 = &b;

if (p1 > p2) cout << p1; //Сравниваются адреса

else cout << p2;

При смешивании в выражении указателей разных типов явное преобразование типов требуется для всех указателей кроме void *.

Указатель неявно может преобразоваться в тип bool (ненулевой указатель – true, нулевой – false).

26 Билет. Функции, объявление функции. Глобальные и локальные переменные в функциях.

Функция - это именованная последовательность описаний и операторов, выполняющих какое-либо законченное действие. Функция может принимать параметры и возвращать значения.

Любая программа на C++ состоит из функций, одна из которых должна иметь имя main(с нее начинается выполнение программы). Функция начинает выполняться в момент вызова. Любая функция должна быть объявлена и определена. Как и для других величин, объявлений может быть несколько, а определение только одно. Объявление функции должно находиться в тексте раньше, чем ее вызова для того, чтобы компилятор мог осуществить проверку правильности вызова.

ОбЪявление функции задает ее имя, тип возвращаемого значения и список передаваемых параметров. Определение функции содержит, кроме объявления, тело функции, представляющее собой последовательность операторов и описаний в фигурных скобках:

[класс] тип имя ([список_параметров])

{ тело функции }

С помощью необязательного модификатора класс можно явно задать область видимости функции, используя ключевые слова extern и static:

extern - глобальная видимость во всех модулях программы(по умолчанию).

static - видимость только в пределах модуля, в которого определена функция.

Тип возвращаемого функцией значения может быть любым, кроме массива и функции(может быть указателем на них). void используют, когда функция не должна ничего возвращать.

Список параметров определяет величины, которые требуется передать в функцию при ее вызове. Элементы списка параметров разделяются запятыми. Для каждого параметра, передаваемого в функцию, указывается его тип и имя.

Все величины, описанные внутри функции, а также ее параметры, являются локальными. Областью их действия является функция. При вызове функции, как и при входе в любой блок, выделяется память под локальные автоматические переменные переменные. ПРи выходе из функции соответствующий участок памяти освобождается, поэтому значение локальных переменных между вызовами одной и той же функции не сохраняются.

Глобальные переменные видны во всех функциях, где не описаны локальные переменные с тем же именем, поэтому использовать их для передачи данных между функциями очень легко.Тем не менее это не рекомендуется, поскольку затрудняет отладку программы и препятствует помещению функции в библиотек иобщего пользования.

27 Билет. Способы передачи параметров в функцию. Использование спецификаторов классов в функциях auto register static extern.

Механизм параметров является основным способом обмена информацией между вызываемой и вызывающей функциями. Параметры, перечисленные в заголовке описания и вызывающей функциями, называются формальными параметрами, или просто параметрами, а записанные в операторе вызова функции - фактическими параметрами, или аргументами.

При вызове функции в первую очередь вычисляются выражения, стоящие на месте аргументов; затем выделяется память под формальные параметры функции в соответствии с их типом, и каждому из них присваивается значение соответствующего аргумента. ПРи этом проверяется соответствие типов и при необходимости выполняется их преобразование.

Существует два способа передачи параметров в функцию: по значению и по адресу.

При передаче по значению в стек заносится копии значений аргументов, и операторы функции работают с этими копиями. Доступ к исходным значениям параметров у функции нет, а следовательно, нет и возможности их изменить.

При по адресу в стек заносится копия адресов аргументов, а функция осуществляет доступ к ячейкам памяти по этим адресам и может изменить исходные значения аргументов.

Чтобы упростить вызов функции, в ее заголовке можно указать значение параметров по умолчанию. Эти параметры должны быть последними с списке и могут опускаться при вызове функции. Если при вызове параметр опущен, должны быть опущены и все параметры, стоящие за ним. В качестве значений параметров по умолчанию могут использоваться константы, глобальные переменные и выражения.

Auto - по умолчанию класс памяти всех локальных переменных объявленных внутри функции. «Время жизни» автоматической переменной ограничено временем выполнения функции, в которой эта переменная определена.

Register - указание компилятору выделить для сохранения данных объекта не ячейку стека, а внутренние регистры процессора. Но это не означает, что компилятор обязательно разместит данные объекта в регистрах процессора. Кроме того, при указании register, компилятор может разместить данные в кэш-памяти. Основной целью объявления переменной (объекта) с ключевым словом register есть обеспечение максимально быстрого доступа к этой переменной и обработки этой переменной. Переменная класса памяти register так же является локальной.

Extern - означает, что переменная определяется в другом месте программы. Используется для создания переменных доступных во всех модулях программы, в которых они объявлены

Static - статическая переменная. Время жизни - постоянное. Инициализируется один раз при первом выполнении оператора, содержащего определение переменной. Переменная с классом памяти static, объявленная внутри функцию будет является локальной, но при этом при повторном запуске этой функции, данные в этой переменной сохраняется.

28 Билет. Понятие перегруженных функций. Понятие прототипа и шаблона функции. Функции с переменным числом параметров.

Использование нескольких функций с одним и тем же именем, но с разными типами параметров, называется перегрузкой функций.

Компилятор определяет, какую именно функцию требуется вызвать по типу фактических параметров. Этот процесс называется разрешением перегрузки. Тип возвращаемого функцией значения в разрешении не участвует.

Если точного соответствия не найдено, выполняются продвижения типов в соответствии с общими правилами. Далее выполняются стандартные преобразования типов. Если соответсвие на одном и том же этапе может быть получено одним и тем же способом, вызов считается неоднозначным и выдается сообщение об ошибке.

Прототип функции (полноценный) состоит из типа возврата функции, её имени и параметров (тип + имя параметра). В кратком прототипе отсутствуют имена параметров функции. Основная часть (между фигурными скобками) опускается. Использование прототипов необходимо, если необходимо вызвать функцию до ее

определения в коде(например в случае, когда функция A вызывает функцию B, а функция B - функцию A).

С помощью шаблона функции можно определить алгоритм, который будет применяться к данным различных типов, а конкретный тип данных передается функции в виде параметра на этапе компиляции. Компилятор автоматически генерирует правильный код, соответствующий переданному типу. Таким образом, создается функция, которая автоматически перегружает сама себя и при этом не содержит накладных расходов, связанных с параметризацией.

Формат простейшей функции-шаблона:

```
template<typename T>
void f(T s)
{
    std::cout << s << '\n';
}

template void f<double>(double); // instantiates f<double>(double)
template void f<>(char); // instantiates f<char>(char), template argument deduced
template void f(int); // instantiates f<int>(int), template argument deduced
```

Если список формальных параметров функции заканчивается многоточием, это означает, что при ее вызове на этом месте можно указать еще несколько параметров. ПРоверка соответствия типов для этих параметров не выполняется. Для доступа к списку параметров используется указатель соответствующего типа. Он устанавливается на начало списка параметров в памяти, а затем перемещается по адресам фактических параметров.

30 Билет. Задачи на формулу Хартли.

Формула Хартли:

$I = \log_2 N$ или $N=2^I$.

N - количество равновероятных исходов события; I – количество бит в сообщении.

1. Какова оптимальная стратегия игры «Угадай число»? Сколько информации можно получить при отгадывании числа из интервала от 1 до 16?

Решение:

Оптимальная стратегия игры строится на получении максимального количества информации тем игроком, который отгадывает, то есть интервал чисел должен всегда делиться пополам, тогда количество чисел в каждом интервале одинаково, а отгадывание интервалов равновероятно. В этом случае на каждом шаге ответ первого игрока («да» или «нет» несет максимальное количество информации).

Таким образом, применяя формулу Хартли, мы получаем

$I = \log_2 16 = 4.$

2. В корзине находятся белые и черные шары. Среди них 18 черных. Сообщение о том, что достали белый шар, несет 2 бита информации. Сколько всего в корзине шаров? (Задача от Поляковой)

Решение:

Вероятность достать белый шар из корзины:

$P_б = \frac{x-18}{x}$, где x – общее количество исходов, P – вероятность события.

Используем формулу Шеннона:

$I = \log_2 \frac{1}{P}$, где I – количество информации о сообщении, P – вероятность события.

Так как нам известна вероятность того, что достали белый шар и известно, сколько это несет информации, то по формуле Шеннона мы можем подсчитать суммарное количество шаров:

$$2 = \log_2 \frac{1}{\frac{x-18}{x}} = \log_2 \frac{x}{x-18}$$

$$4 = \frac{x}{x-18}$$

$$4(x-18) = x$$

$$4x - x = 72$$

$$x = 24.$$

3. В велокроссе участвуют 119 спортсменов. Специальное устройство регистрирует прохождение каждым из участников промежуточного финиша , записывая его номер с использованием минимально возможного количества бит, одинакового для каждого спортсмена. Каков информационный объем сообщения, записанного устройством, после того, как промежуточный финиш прошли 70 велосипедистов?

Решение:

$$I = \log_2 N = \log_2 119 = 6{,}8948$$

– количество бит, необходимых для записи одного участника. Тогда для записи информации о 70 велосипедистах потребуется 482,636 бит.

31 Билет. Задачи с флажками.

Флаг — это полотнище правильной (как правило, прямоугольной) формы прикрепленное к древку или поднимаемое на специальной мачте (флагштоке). Исторически флаги появились для передачи простых сигналов на поле боя. Например: подняли флаг, и конница понеслась в атаку! В простейшем случае с помощью флага передается информация объемом 1 бит (одно из двух: флаг поднят или нет).

Переменная-флаг — это, как правило, переменная логического типа, значение которой сигнализирует о состоянии вычислительного процесса. Приведем несколько примеров, когда результат может характеризоваться всего одной логической переменной:

1. Подводится баланс коммерческого предприятия. Дальнейшие действия могут зависеть от того, будет он положительным или отрицательным. Если отрицательный, надо просить кредит, положительный — планировать отдых в Крыму. В общем, самая существенная информация может быть передана одним битом.

2. Решаем квадратное уравнение. Если дискриминант не отрицательный — ищем корни. Для хода вычислительного процесса важен факт не отрицательности, который также содержит 1 бит информации и может, таким образом, быть сохранен с помощью логической переменной.

3. Детям на уроке физкультуры велено построиться по росту. Если они построились не по росту, надо на них наорать. Опять действия учителя зависят от информации объемом 1 бит.

Но кто и кому может передавать информацию в ходе выполнения программы? Дело в том, что при разработке больших программ происходит разделение задачи на более мелкие подзадачи (блоки), каждая из которых решается отдельно и, может быть даже, разными людьми. В этом случае один блок, закончив свою работу должен передать ее результат другому блоку. Здесь и могут пригодиться флаги.

Не обязательно использовать в качестве флага именно логическую переменную. В принципе флагом может считаться любая переменная, принимающая небольшое количество возможных значений, каждое из которых характеризует тот или иной результат вычислительного процесса.

Пример использования флага

Задача — проверка упорядоченности последовательности.

Пользователь вводит последовательность чисел, завершающуюся нулем. Требуется проверить, упорядочены ли они по возрастанию.

```
Growing = True  #Переменная флаг
x = float(input())
old_x = x-1 #будем хранить предыдущий член последовательности,
#в первый раз инициализированный так, чтобы не загубить переменную-
флаг
while x != 0:
    Growing = Growing and (x > old_x)
    old_x = x
    x = float(input())
if Growing:
    print('последовательность возрастающая')
```

else:

print('последовательность не является монотонно возрастающей')

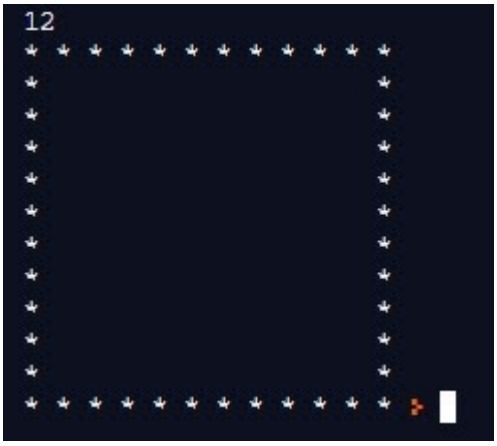
Если очередное введенное число (x) будет меньше предыдущего (old_x), то флаг Growing примет значение False и сохранит это значение до конца цикла.

```
#include <iostream>
#include <string>
#include <windows.h>
using namespace std;
int main()
{
	SetConsoleOutputCP(CP_UTF8);
	int a, b;
	cin >> a;
	if (a != 0) {
		cin >> b;
		bool f = true;
		while (a < b && f == true) {
			cin >> b;
			f = f && a < b;
			if (f == true) {
				a = b;
			}
		}
	}
	return 0;
}
```

32 Билет. Задачи на фигуры(треугольники квадраты и т.д.).

```cpp
//Программа печатает пустой квадрат со стороной N.
int n; //N = длина стороны квадрата. N - длина строки и следовательно количество звёздочек в строке

int main()
{
  cin >> n;
  for (int i = 1; i <= n; i++) // Этот и последний цикл выводят N звёздочек (цикл включает в себя элементы от 1 до n включительно, так как N - длина строки и следовательно количество звёздочек в строке) - длину стороны квадрата. Пробелы для красоты.
  {
    cout << "* ";
  }
  cout << endl;
  for (int i = 1; i <= n-2; i++) //Выводятся N-2 строчки - так как у нас два цикла, которые выводят в итоге 2 строчки со сторонами квадрата
  {
    cout << "* "; //Первый символ
    for (int j = 1; j <= n-2; j++)
    {
      cout << "  "; //В этом (вложенном) цикле выводятся J-2 пробелов, так как у нас первый и последний символы в строчке будут звёздочками
    }
    cout << "* "; //Последний символ
    cout << endl;
  }
  for (int i = 1; i <= n; i++) //Последний цикл
  {
    cout << "* ";
  }
  return 0;
}
```



33 Билет. Задачи на методы решения нелинейных уравнений.

1) **Метод последовательных приближений** (метод итераций)

Реализация на C++ для рассмотренного выше примера

$$\sin(x) = \frac{1}{x}$$

Уравнение может быть записано в форме

$$x = \frac{1}{sin\left(\frac{\pi}{180}x\right)}$$

```
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
using namespace std;
double find(double x, double eps)
{
  double rez; int iter = 0;
  cout << "x0= " << x << " ";
  do {
    rez = x;
    x = 1 / (sin(M_PI*x / 180));
    iter++;
  } while (fabs(rez - x) > eps && iter<20000);
  cout << iter << " iterations" << endl;
  return x;
}
int main()
{
  cout << find(7, 0.00001);
  cin.get();
  return 0;
}
```

2) **Метод Ньютона** (метод касательных)

Для заданного уравнения

$$f(x) = sin\left(\frac{\pi}{180}x\right) - \frac{1}{x} = 0$$

производная будет иметь вид $$f'(x) = \frac{\pi}{180}cos\left(\frac{\pi}{180}x\right) + \frac{1}{x^2}$$

```
#define _USE_MATH_DEFINES
#include <iostream>
#include <cmath>
using namespace std;
double find(double x, double eps)
{
  double f, df; int iter = 0;
  cout << "x0= " << x << " ";
  do {
    f = sin(M_PI*x / 180) - 1 / x;
    df = M_PI / 180 * cos(M_PI*x / 180) + 1 / (x*x);
    x = x - f / df;
    iter++;
  } while (fabs(f) > eps && iter<20000);
  cout << iter << " iterations" << endl;
  return x;
}
int main()
{
  cout << find(1, 0.00001);
  cin.get(); return 0;
}
```

3) Метод половинного деления

Пусть уравнение $F(x) = 0$ имеет один корень на отрезке $[a;b]$. Функция $F(x)$ непрерывна на отрезке $[a; b]$.
Метод половинного деления заключается в следующем: Сначала выбираем начальное приближение, деля отрезок пополам, т.е. $x0 = (a+b)/2$.
Если $F(x) = 0$, то $x0$ является корнем уравнения. Если $F(x) \neq 0$, то выбираем тот из отрезков, на концах которого функция имеет противоположные знаки. Полученный отрезок снова делим пополам и выполняем действия сначала и т.д

Процесс деления отрезка продолжаем до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданного числа .

34 Билет. Задачи на последовательности нахождение заданного элемента максимального и минимального и др.

Вводим N, которое отвечает за количество чисел в последовательности, затем объявляем цикл (for от 1 до N), в котором поочерёдно вводим числа. Первое число присваивается максимуму и минимуму, и уже следующие введённые числа сравниваются с ними.

```
cin >> n;
for (int i = 1; i <= n; i++)
{
    cin >> t;
    if (t > max) max = t;
    if (t < min) min = t;
}
cout << "min = " << min;
cout << "\nmax = " << max;
```

При N = 5 и вводе последовательности {1, 2, 3, 4, 5} программа выведет:

```
5
1 2 3 4 5
min = 1
max = 5
```

35 Билет. Задачи на вложенные циклы.

Напечатать прямоугольный равнобедренный треугольник со сторонами катетов, равными N, и прямым углом справа снизу, где N — натуральное число больше 2.

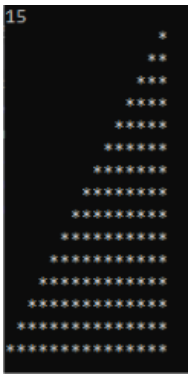
```
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        for (int j = i; j < n - 1; j++)
        {
            cout << " ";
        }
        for (int j = 0; j <= i; j++)
        {
            cout << "*";
        }
        cout << "\n";
    }
    return 0;
}
```

В основном цикле for, цикл включает в себя элементы от 1 до n включительно (так как n это катет данного треугольника и следовательно количество строк). Потом во вложенном цикле for (выводим пробелы), цикл включает в себя элементы от 1 до n-1 включительно (так как прямой угол находится справа снизу). Потом во вложенном цикле for (выводим звёздочки), цикл включает в себя элементы от 1 до i включительно, где i-элемент основного цикла. Потом переходим на следующую строку.

При N = 4 программа выдаёт:

```
4
   *
  **
 ***
****
```

При N = 15 программа выдаёт:



36 Билет. Задачи на работу с массивам.

Массив - это структура данных, представленная в виде группы ячеек одного типа, объединенных под одним единым именем. Массивы используются для обработки большого количества однотипных данных. Имя массива является указателем. Отдельная ячейка данных массива называется элементом массива. Элементами массива могут быть данные любого типа. Массивы могут иметь как одно, так и более одного измерений. В зависимости от количества измерений массивы делятся на одномерные массивы, двумерные массивы, трёхмерные массивы и так далее до n-мерного массива. Чаще всего в программировании используются одномерные и двумерные массивы

Int a[n]; - Синтаксис объявления одномерного массива в C++, где int – тип данных элементов массива, a – название массива, n – размер массива (кол-во переменных, которые могут в нем хранится).

Количество измерений массива изменяется в зависимости от кол-ва заданных размеров в квадратных скобках, например:

Int a[n][m] – двумерный массив, int a[n][m][k] – трёхмерный и т.д.

Int a[10] = {5, -12, -12, 9, 10, 0, -9, -12, -1, 23}; - Так можно собственноручно задать значения элементов в массиве.

С массивами в основном работают с помощью цикла for:

```
int main(){
    int n;              //Переменная для размера массива
    cin >> n;           //Ввод размера массива
    int mas[n];         //Обьявление массива
    for(int i=0; i<n; i++){   //Ввод массива через цикл for с вводом элементов через консоль
        cin >> mas[i];
    }
    for(int i=0; i<n; i++){    //Вывод введенного массива
        cout << mas[i] << " ";

    }
}
```

Любые другие операции, также делаются в основном через for, т.к. мы проходим от i=0 (т.е. первой переменной в массиве), до n(Последней переменной).

37 Билет. Методы сортировки массивов теория и задачи.

Сортировка методом простого выбора(selection):

Выбирается минимальный элемент массива и меняется местами с первым элементом массива. Затем процесс повторяется с оставшимися элементами и т. д.

```
for (int i = 0; i < size - 1; i++)
{
  min = i;
  for (int j = i + 1; j < size; j++)
  {
    if (arr[j] < arr[min])
      min = j;
  }
  temp = arr[i];
  arr[i] = arr[min];
  arr[min] = temp;
}
```

Size – размер массива;

Arr – название массива;

Temp – переменная для перемещения элементов между собой;

Min – индекс минимального элемента в массиве;

Сортировка методом простого обмена(bubble или пузырек):

Сравниваются и меняются местами пары элементов, начиная с последнего. В результате самый маленький элемент массива оказывается самым левым элементом массива. Процесс повторяется с оставшимися элементами массива.

```
for (int i = 0; i < size - 1; i++) {
    for (int j = 0; j < size - i - 1; j++) {
        if (arr[j] > arr[j + 1]) {
            temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
    }
}
```

Size – размер массива;

Arr – название массива;

Temp – переменная для перемещения элементов между собой;

Сортировка с помощью включения(inclusion,insertion):

Элементы массива делятся на уже готовую последовательность и исходную. При каждом шаге, начиная с I=2, из исходной последовательности извлекается i-ый элемент и вставляется на нужное место готовой последовательности, затем i увеличивается на 1 и т. д.

В процессе поиска нужного места осуществляются пересылки элементов больше выбранного на одну позицию вправо, т. е. выбранный элемент сравнивают с очередным элементом отсортированной части, начиная с j:=i-1. Если выбранный элемент больше a[i], то его включают в отсортированную часть, в противном случае a[j] сдвигают на одну позицию, а выбранный элемент сравнивают со следующим элементом

отсортированной последовательности. Процесс поиска подходящего места заканчивается при двух различных условиях:

• если найден элемент a[j]>a[i];

• достигнут левый конец готовой последовательности.

```
for (int i = 1; i < size; i++)
{
  int value = arr[i];
  int index = i;
  while ((index > 0) && (arr[index - 1] > value))
  {
    arr[index] = arr[index - 1];
    index--;
  arr[index] = value;
  }
}
```

Size – размер массива;

Arr – название массива;

Value – запоминает значение элемента;

Index – запоминает индекс элемента;

38 Билет. Задачи на двумерные массивы.

Двумерный массив - это одномерный массив, элементами которого являются одномерные массивы. Другими словами, это набор однотипных данных, имеющий общее имя, доступ к элементам которого осуществляется по двум индексам. Наглядно двумерный массив удобно представлять в виде таблицы, в которой n строк и m столбцов, а под ячейкой таблицы, стоящей в i-й строке и j-м столбце понимают некоторый элемент массива a[i][j].

n-мерный массив - это одномерный массив, элементами которого являются (n-1)-мерные массивы.

```
//определение двумерного массива
int a[50][50];
int n, m;
cin » n;
cin » m;
//ввод массива
for (int i = 0; i < n; i++)
{
for (int j = 0; j < m; j++)
{
cin » a[i][j];
}
}
//находим среднее арифметическое
float s = 0;
int k = 0;
for (int i = 0; i < n; i++) {
for (int j = 0; j < m; j++) {
```

```
s += a[i][j];
k++;
}
}
cout « "среднее арифметическое = ";
cout « s / k;
```

39 Билет.Задачи по работе с функциями (перегруженными и с переменным числом параметров).

Перегруженные функции должны находиться в одной области видимости, иначе произойдет сокрытие аналогично одинаковым именам переменных во вложенных блоках.

Перегруженные функции могут иметь параметры по умолчанию, при этом значения одного и того же параметра в разных функциях должны совпадать. В различных вариантах перегруженных функций может быть различное количество параметров по умолчанию.

Функции не могут быть перегружены, если описание их параметров отличается только модификаторами const, volatile или использованием ссылки (например, int и const int или int и int&).

Нельзя перегружать функции, отличающиеся только типом возвращаемого значения.

Ничего сложного в механизме перегруженных функций нет. Перегруженные функции являются, по сути, совершенно различными функциями, идентифицируемыми не только своим именем (оно у них одно и то же), но и списком параметров. Компилятор выполняет т. н. декорирование имен. дополняя имя функции кодовой последовательностью символов, кодирующей тип ее параметров. Тем самым формируется уникальное внутреннее имя.

Вот несколько примеров того, как для различных прототипов производится декорирование имени функции:

void Func(void); // @Func$qv

void Func(int); // @Func$qi

void Func(int, int); // @Func$qii

void Func(*char); // @Func$qpc

void Func(unsigned); // @Func$qui

void Func(const char*); // @Func$qpxc

- Тип возвращаемого значения никак не отражается на декорировании имени.
- Код C++ Функции с произвольным числом параметров переменной длины

```
#include <stdlib.h>
#include <iostream.h>
#include <stdarg.h>

void OK(char *format, …) //Функция OK с произвольным числом параметров
{
va_list ap; //Указатель на список параметров
va_start(ap,format); //Настроились на список параметров
for (char *p=format;*p;p++) //
{
if (*p==’%‘) //Если встретится символ %
{
switch (*++p) //То анализируем следующий за этим симолом символ
{
case ‘d‘: int ival=va_arg(ap,int); //Если это символ d, то значит параметр int
cout«ival«” “;break; //Выводим параметр типа int на экран
```

```
case ‘f‘: double dval=va_arg(ap,double); //Если это символ f значит параметр double
cout«dval«” “;break; //Выводим параметр типа double на экран
}
}
else cout«*p«” “;
}
va_end(ap); //Завершаем работу с макрокомандами
}

void main()
{
system(“CLS”);

OK(“%d%f“,8,9.555); //Вывод двух чисел с разными типами
system(“PAUSE”);
return;
}
```

специальный тип va_list используется для представления списков параметров неизвестной длины и состава

va_start вызывается непосредственно перед началом работы с неименованными параметрами

ap инициализируется указателем на последний именованный параметр в списке с переменным числом параметров — “параметр” lastarg

Здесь под параметр является условным обозначением и относится к макрокомандам, а не функциям

После вызова макроопределения va_start, каждый вызов va_arg возвращает значение заказанного типа type себе. Надо заранее указать тип желаемого параметра. В некоторых реализациях с макроопределением va_arg запрещено использовать типы char, unsigned char, float. Даже в макроопределениях, предназначенных для стандартизации языка (речь идет не о Borland C++ 3.1, а о более новых), многое зависит от реализации.

Работа со списком параметров завершается вызовом макроопределения void va_end(ap); Это макроопределение обеспечивает корректный возврат из функции и вызывается перед выходом из функции.