Вопросы к экзамену по модулю ООП. Зима 2024-25

1. Язык. Алфавит. Классификации языков программирования по уровню, по типизации, по типу трансляции.  
     
   Ответ:  
     
   Алфавит – это набор символов, используемых для написания программ  
   Классификация по уровню:   
   низкий – небольшой разброс семантики и прагматики,   
   высокий - большой разброс семантики и прагматики,   
   сверхвысокий – очень большой разброс семантики и прагматики  
   Классификация по типизации: статический, динамический  
   Классификация по типу трансляции: компилируемый, интерпретируемый
2. Язык. Алфавит. Классификации языков программирования по подходу.  
     
   Ответ:  
     
   Алфавит – это набор символов, используемых для написания программ  
     
   Классификация по подходу:   
   Линейная парадигма – имеет последовательный вид  
   Блочная парадигма – программа разбивается на блоки (циклы, ветвление)  
   Процедурно-ориентированная – когда в программу добавляются вспомогательные алгоритмы  
   Функциональная парадигма – функция может восприниматься как единица данных  
   Структурно - ориентированная – это подход к программированию, где данные могут формировать структуры  
   ООП – это подход к программированию, где данные могут формировать объекты  
   Аспектно-ориентированная - это подход к программированию, где основной единицей данных является аспект
3. Объектно-ориентированное программирование. Основные определения. Виды отношений между классами. Примеры.  
     
   Ответ:  
     
   Объект – это мнимая или реальная сущность, обладающая конкретными свойствами и поведением.  
   Свойства называют еще полями, атрибутами, параметрами.  
   Поведение называют методами.  
   Класс – это шаблон поведения и свойств объектов.  
   Метакласс – это шаблон свойств и поведения класса.  
   Инкапсуляция – это процесс сокрытия внутреннего представления от внешнего.  
   Полиморфизм – это побочный продукт наследования.  
     
   Отношения между классами:  
   Агрегация – это вид взаимодействия, при котором 1 из классов полностью владеет другим (1 - ∞).  
   Ассоциация – это вид взаимодействия, при котором ни у 1 из классов нет выраженного характера владения над другим.  
   Наследование – это вид отношения, при котором класс частично или полностью перенимает свойства и поведения другого.  
   Реализация – это вид отношения между классами, при котором 1 из классов реализует абстрактное поведение другого.  
   Зависимость – это вид отношений между классами, при котором 1 класс зависит от другого.  
   Композиция – это вид отношений между классами, при котором 1 из классов является частью другого и не может функционировать без него.
4. Лексика языков C/C++. Идентификаторы, литералы, разделители, операторы, ключевые слова. Примеры.  
     
   Ответ:  
     
   Идентификаторы (Имя) - это последовательность символов, используемая для обозначения одного из следующих элементов.  
   1. a-z, A-Z, 0-9, \_ (63 символа)

2. Не может начинаться с цифры (компьютер читает последовательно, компиляторы используют префиксный анализ)  
Литералы – это лексема описывающее значение, существуют целочисленные, дробные, символьные, строковые, логические и Нул-литерал.  
Разделители (пробел, таб, \n, (), {}, ‘,’, ‘;’), пробел, таб и \n для компилятора являются одним и тем же  
Операторы могут быть задающими, арифметическими, сравнительными, логическими, битовыми, адресными(&, \*, ‘.’, ->, []) и т. д.  
Ключевыми словами являются: if, for, while, do, break, continue, все типы данных и др.

1. Синтаксис языков C/C++. Унарные, бинарные, тернарные и сложные операторы (условий, выбора, циклов). Примеры.  
     
   Ответ:  
   Унарные операторы – это операторы, совершающие действие над 1 операндом и возвращающим 1 результат (+, -, \*, &, ~, !).  
   Бинарные операторы - оператор, принимающий два аргумента и возвращающий один результат (все задающие операторы за исключением инкремента и декремента, &&, ||)  
   Тернарные операторы - операторы совершающие действие над 3 операндом и возвращающим 1 результат (тернарный оператор в C++)  
   Cложные операторы – операторы, совершающие действие над более чем 3 операндами(if, switchcase, while, for)
2. Синтаксис языков C/C++. Функции. Примеры.

Ответ:

Функция — фрагмент программного кода, к которому можно обратиться из другого места программы. Функция может принимать параметры и должна возвращать некоторое значение, возможно пустое. Функции, которые возвращают пустое значение, часто называют процедурами. Объявление функции, кроме имени, содержит список имён и типов передаваемых параметров (или: аргументов), а также, тип возвращаемого функцией значения. В объектно-ориентированном программировании функции, объявления которых являются неотъемлемой частью определения класса, называются методами. Функция определяет собственную (локальную) область видимости, куда входят входные параметры, а также те переменные, которые объявляются непосредственно в теле самой функции.  
  
Пример: функция модуля abs(), максимум из 2 элементов max()

1. Синтаксис языков C/C++. Классы. Примеры.  
     
   Ответ:  
     
   Класс – это шаблон поведения и свойств объектов.

Мета класс – это шаблон свойств и поведения класса. Класс может содержать внутри себя поля и методы (функции внутри класса), Класс имеет внутри себя модификаторы доступа: public, private и protected.  
public – доступ имеют все  
protected – доступ имеет класс, а также все в нем внутри содержимое (методы, поля, классы, которые композируют и наследуют его)  
private - доступ имеет класс, а также все в нем внутри содержимое (методы, поля, классы, которые композируют его)  
  
Инкапсуляция – процесс сокрытия внутреннего представления от внешнего.  
Наследование – это вид отношения, при котором класс частично или полностью перенимает свойства и поведения другого.   
Полиморфизм – это побочный продукт наследования.  
  
Класс также имеет внутри себя конструктор и деструктор.  
Конструктор – это специальный метод класса, вызываемый непосредственно после выделения оперативной памяти.  
Деструктор - это специальный метод класса, вызываемый непосредственно перед удалением оперативной памяти.

1. Статическая и динамическая память. Работа с динамической памятью в языке C.  
     
   Ответ:  
     
   Память делится на статическую и динамическую. Динамическая память – это память, которая выдается во время исполнения программы. Библиотека stdlib имеет функции для работы с дин. памятью:  
   void \*malloc(размер ячейки)  
   void \*calloc(кол-во ячеек, размер 1 ячейки)  
   void \*realloc(указатель на старую память, размер новой)  
   void \*free(указатель)
2. Статическая и динамическая память. Работа с динамической памятью в языке C++.  
     
   Ответ:  
     
   Память делится на статическую и динамическую. Динамическая память – это память, которая выдается во время исполнения программы. С++ использует новые методы работы с динамической памятью при помощи операторов new и delete:  
   new – выделение памяти  
   delete – освобождение памяти
3. Массивы в языках C/C++. Принципы хранения и работы.  
     
   Ответ:  
     
   Массив - это упорядоченные ячейки памяти одного типа, к которым можно обращаться по индексу. В C++ существуют вектора, реализующие структуру массива с дополнительными функциями. Элементы массива хранятся в смежных адресах памяти в порядке увеличения от первого элемента к последнему.
4. Система ввода/вывода в языке C. Стандартные потоки ввода. Файлы. Функции взаимодействия с потоками ввода. Примеры.  
     
   Ответ:  
     
   Файл состоит из хранилища, в котором он лежит.  
   Буфер – область памяти для временного хранения или трансфера данных. В языке C существуют функции для работы с буфером:  
   setbuf() – задает буфер, устанавливает внутренний буфер для использования в потоковых операциях  
   setvbuf() - задает буфер по-другому, изменяет режим буферизации указанного потока файла  
   fopen() – открывает файл  
   fclose() – закрывает файл  
   fflush() – для сброса буфера  
   freopen() – пере открывает файл
5. Система ввода/вывода в языке C++. Стандартные потоки ввода. Файлы. Функции, классы и объекты взаимодействия с потоками ввода. Примеры.  
     
   Ответ:  
     
   Библиотека iostream  
   cout используется для вывода и cin для ввода, используя базовые классы basic\_ostream и basic\_istream. Операцию cerr используют для показа ошибок, в нем выключена буферизация.
6. Строки в языке C. Функции работы со строками. Примеры.  
     
   Ответ:  
     
   Строковый тип данных в C как таковой отсутствует, а в качестве строк в С используются обычные массивы символов. Так как строки на языке С являются массивами символов, то к любому символу строки можно обратиться по его индексу. Для этого используется синтаксис обращения к элементу массива, поэтому первый символ в строке имеет индекс ноль. Например, в следующем фрагменте программы в строке str осуществляется замена всех символов 'a' на символы 'A' и наоборот.

for(int i = 0; str[i] != 0; i++)

{

if (str[i] == 'a') str[i] = 'A';

else if (str[i] == 'A') str[i] = 'a';

}  
  
Для ввода и вывода строковой информации можно использовать функции форматированного ввода и вывода (printf и scanf). Для этого в строке формата при вводе или выводе строковой переменной необходимо указать спецификатор типа %s. Например, ввод и последующий вывод строковой переменной будет иметь вид:

char str[31] = "";

printf("Введите строку: ");

scanf("%30s”,str);

printf("Вы ввели: %s”,str);

1. Строки в языке C++. Функции, классы и объекты работы со строками. Примеры.  
     
   Ответ:  
     
   Библиотека string предоставляет удобные и мощные инструменты для работы со строками в языке программирования C++. Она позволяет легко создавать, изменять и преобразовывать строки. Для ввода строк с клавиатуры в C++, вы можете использовать объект класса getline из библиотеки <string>.  
     
   Пример кода для ввода строки и вывода ее на экран

#include <iostream>

#include <string>

using namespace std;

int main()

{

string input;

getline(cin, input);

cout << input << endl;

return 0;

}

1. Создание классов в языке C++. Методы инкапсуляции. Конструкторы и деструкторы классов в языке C++. Примеры.  
     
   Ответ:  
     
   Инкапсуляция – процесс сокрытия внутреннего представления от внешнего.  
     
   public – доступ имеют все

protected – доступ имеет класс, а также все в нем внутри содержимое (методы, поля, классы, которые композируют и наследуют его)

private - доступ имеет класс, а также все в нем внутри содержимое (методы, поля, классы, которые композируют его)  
  
Класс также имеет внутри себя конструктор и деструктор.  
Конструктор – это специальный метод класса, вызываемый непосредственно после выделения оперативной памяти.  
Деструктор - это специальный метод класса, вызываемый непосредственно перед удалением оперативной памяти.  
  
Пример:  
  
class date

{

int day, year;

char \*month;

public:

date(int d, char\* m, int y)

{

day = d;

month = new char[strlen(m)+1];

strcpy\_s(month, strlen(m)+1,m);

year = y;

}

~date() { delete[] month; } // деструктор

};

1. Создание классов в языке C++. Перегрузка операторов в языке C++. Дружественные операторы. Примеры.  
     
   Ответ:  
     
   Перегрузка операторов позволяет определить для объектов классов втроенные операторы, такие как +, -, \* и т.д. Для определения оператора для объектов своего класса, необходимо определить функцию, название которой содержит слово operator и символ перегружаемого оператора. Функция оператора может быть определена как член класса, либо вне класса.  
   Пример перегрузки:  
   class Counter

{

public:

Counter(int val)

{

value =val;

}

void print()

{

std::cout << "Value: " << value << std::endl;

}

Counter operator + (const Counter& counter) const

{

return Counter{value + counter.value};

}

private:

int value;

};  
  
friend – дружественная функция, которая находится вне класса, но при этом имеет доступ к его содержимому в private.  
  
Пример friend функций:  
  
class Auto

{

friend void drive(const Auto&);

friend void setPrice(Auto&, unsigned);

public:

Auto(std::string autoName, unsigned autoPrice)

{

name = autoName;

price = autoPrice;

}

void print()

{

std::cout << name << " : " << price << std::endl;

}

private:

std::string name; // название автомобиля

unsigned price; // цена автомобиля

};

void drive(const Auto &car)

{

std::cout << car.name << " is driven" << std::endl;

}

void setPrice(Auto &car, unsigned price)

{

car.price = price;

}

1. Шаблонизирование функций и классов.  
     
   Ответ:  
     
   Шаблон – это средство языка C++, предназначенное для кодирования обобщённых алгоритмов.  
   Пример шаблонизирования:  
     
   template< typename T >

void sort( T array[], int size ) // объявление и определение

{

T t;

for (int i = 0; i < size - 1; i++)

for (int j = size - 1; j > i; j--)

if (array[j] < array[j-1])

{

t = array[j];

array[j] = array[j-1];

array[j-1] = t;

}

}  
  
В классе, реализующем связный список целых чисел, алгоритмы добавления нового элемента списка, поиска нужного элемента не зависят от того, что элементы списка — целые числа. Те же алгоритмы применялись бы и для списка символов, строк, дат, классов игроков и так далее.

template< class T >

class List

{

/\* ... \*/

public:

void Add( const T& Element );

bool Find( const T& Element );

/\* ... \*/

};

1. Связные списки. Хранение. Реализация. Примеры.  
     
   Ответ:  
     
   Связный список — структура данных, состоящая из элементов, содержащих помимо собственных данных ссылки на следующий и/или предыдущий элемент списка.  
     
   #ifndef LIST\_H

#define LIST\_H

#include <cstdlib>

#include <iostream>

#include <iterator>

template<typename ElementT>

class List {

private:

struct Node {

ElementT value;

List<ElementT>::Node \*next;

List<ElementT>::Node \*prev;

Node( // c++ < 20

const ElementT &value,

List<ElementT>::Node \*next,

List<ElementT>::Node \*prev

);

};

List<ElementT>::Node \*head;

List<ElementT>::Node \*tail;

std::size\_t size;

public:

List();

~List();

void push\_back(const ElementT &element);

ElementT &back();

void push\_front(const ElementT &element);

ElementT &front();

template<typename Char\_t, typename Element\_t>

friend std::basic\_ostream<Char\_t> &operator<<(std::basic\_ostream<Char\_t> &out, List<Element\_t> &list);

struct iterator {

private:

using node\_pointer = Node \*;

node\_pointer m\_ptr;

public:

using iterator\_category = std::forward\_iterator\_tag;

using difference\_type = std::ptrdiff\_t;

using value\_type = ElementT;

using pointer = value\_type \*;

using reference = value\_type &;

iterator(node\_pointer ptr);

reference operator\*();

pointer operator->();

iterator &operator++();

friend bool operator==(const iterator &a, const iterator &b) {

return a.m\_ptr == b.m\_ptr;

}

friend bool operator!=(const iterator &a, const iterator &b) {

return a.m\_ptr != b.m\_ptr;

}

};

iterator begin();

iterator end();

};

#include "List.tpp"

#endif

1. Коллекции C++. Итераторы. Виды и принцип работы.  
     
   Ответ:  
     
   Итераторы — это обобщение указателей, позволяющее программе на C++ работать с различными структурами данных единообразно.  
     
   Input Iterator (входной итератор)

Предназначен для последовательного чтения элементов.

Пример: чтение данных из файла или потока.

Операции: \* (разыменование), ++ (переход к следующему элементу).

Пример контейнеров: istream\_iterator.  
  
Output Iterator (выходной итератор)

Предназначен для последовательной записи данных.

Пример: запись данных в файл или поток.

Операции: = (присвоение), ++ (переход к следующей позиции).

Пример контейнеров: ostream\_iterator.

Forward Iterator (прямой итератор)

Поддерживает чтение и запись элементов, движение только вперёд.

Пример контейнеров: forward\_list.

Операции: \*, ++.

Bidirectional Iterator (двунаправленный итератор)

Поддерживает чтение и запись, движение вперёд и назад.

Пример контейнеров: list, set, map.

Операции: \*, ++, --.

Random Access Iterator (итератор с произвольным доступом)

Поддерживает доступ к любому элементу с использованием арифметики указателей.

Пример контейнеров: vector, deque, массивы.

Операции: \*, ++, --, +n, -n, [].  
  
Legacy Contiguous Iterator(итератор с непрерывным

хранением)  
Поддерживает непрерывное хранение.

1. Коллекции C++. array. vector. string. Принцип работы.  
     
   Ответ:  
     
   Массив - это упорядоченные ячейки памяти одного типа, к которым можно обращаться по индексу. В C++ существуют вектора, реализующие структуру массива с дополнительными функциями, а также в библиотеке string есть реализация массива char (строки) с дополнительными функциями.
2. Коллекции C++. list. forward\_list. deque. Принцип работы.  
     
   Ответ:  
     
   Связный список(list) — структура данных, состоящая из элементов, содержащих помимо собственных данных ссылки на следующий и/или предыдущий элемент списка.  
     
   forward\_list в языке программирования C++ представляет собой односвязный список, который обеспечивает эффективные операции вставки и удаления элементов в начале списка. Этот контейнер входит в Standard Template Library (STL) и предоставляет компактный и маневренный интерфейс для управления данными.  
     
   Двусторонняя очередь (deque) — абстрактный тип данных, в котором элементы можно добавлять и удалять как в начало, так и в конец. Может быть реализована при помощи двусвязного списка.
3. Функциональное программирование. Функтор. Предикат. Примеры.  
     
   Ответ:  
     
   Функторы и предикаты – это подход к программированию.  
   Функтор – объект, который может быть воспринят как функция.  
   Предикат – частный случай функтора, когда возвращается булевское значение.  
     
   Пример функтора:

template <int k, int b>  
int f (int x)

{

return k\* x + b;  
}  
  
auto f1 = f<1, 1>;  
auto f2 = f<2, 3>;  
class F  
{  
private:  
 double k, b;  
public:  
 F(double k, double b) : k(k), b(b) {}  
 double calc(double x) return k\*x + b;  
 double operator() (double x) return calc(x);  
};  
  
Пример предиката:  
  
class DivFilter  
{  
private:  
 int divider;  
public:  
 DivFilter(int divider) : divider(divider) {}  
 bool operator () (int value){  
 return value % divider = = 0;  
 }  
};

1. Функциональное программирование. Функция как параметр функции.   
     
   Ответ:  
     
   Функция может также служить и одним из параметров другой функции, в которой может вычисляться часть ответа.

Пример:  
def F(x):

return x \* 2  
def Res(x, kakayato\_funkciya):  
 return kakayato\_funkciya(x) + x  
  
n = int(input())  
print(Res(n, F))

1. Функциональное программирование. Анонимные функции.  
     
   Ответ:  
     
   Анонимная функция — особый вид функций, которые объявляются в месте использования и не получают уникального идентификатора для доступа к ним. При помощи такой функции код становится более гибким и более экономным. Анонимные функции используют тогда, когда она нам нужна за всю программу 1 раз.  
     
   [захват](параметры)mutable исключения атрибуты->возвращаемый\_тип{тело}  
     
   Пример:

[&y](int x){ return x + y; }

1. Функциональное программирование. Неограниченное количество параметров функции  
     
   Ответ:

Функции, в объявлениях которых в качестве последнего члена указано многоточие (...), могут принимать переменное число аргументов. В таких случаях C++ обеспечивает проверку типа только для явно объявленных аргументов. Если va\_list экземпляр создается, передается в другую функцию и используется через va\_arg в этой функции, то любому последующему использованию в вызывающей функции должен предшествовать вызов va\_end .  
Методы с va\_list:

va\_start - обеспечивает доступ к аргументам вариативной функции  
va\_copy - создает копию аргументов вариативной функции

va\_arg - получает доступ к следующему аргументу вариативной функции  
va\_end - завершает обход аргументов вариативной функции

Пример функции с вариативным количеством параметров:  
  
double average(int count, ...)

{

va\_list ap;

int j;

double sum = 0;

va\_start(ap, count);

for (j = 0; j < count; j++)  
 {

sum += va\_arg(ap, double);

}

va\_end(ap);

return sum / count;

}