1. Static variable
2. Static functions in class
3. Vector
4. Stod
5. Type size
6. Pointer vs reference
7. array
8. Map
9. Unorderd\_map

Map vs unordered\_map

1. Set
2. Deque
3. Queue
4. Double ended queue
5. Stack
6. Object Oriented Programming

Polymorphism

1. Polymorphism

<https://www.geeksforgeeks.org/polymorphism-in-c/>

https://www.geeksforgeeks.org/polymorphism-in-c/

Function Overloading

Operator overloading

Function overriding

1. Virtual

Virtual variable

Virtual function

1. Constructor

Overloading constructor

1. Destructor
2. Function overloading
3. Bit calculate
4. Template
5. Compilation
   1. Preprocessing
   2. Compilation
   3. linking
6. access specifier
   1. public
   2. protected
   3. private
7. tree:

in-order: left -> node -> right

pre-order: node->left->right

post-order: left->right->node

1. data structures / struct

Static Variables:

Variables in a function, Variables in a class

Static Members of Class : Class objects and Functions in a class

Static variables in a class: As the variables declared as static are initialized only once as they are allocated space in separate static storage so,

the static variables in a class are shared by the objects. There can not be multiple copies of same static variables for different objects.

Also because of this reason static variables can not be initialized using constructors.

Static functions in a class:

just like the static data members or static variables inside the class,

static member functions also does not depend on object of class. We are allowed to invoke a static member function using the object and the ‘.’ operator but it is recommended to invoke the static members using the class name and the scope resolution operator.

Static member functions are allowed to access only the static data members or other static member functions,

they can not access the non-static data members or member functions of the class.

Vector

Vectors are sequence containers representing arrays that can change in size.

It is a dynamically allocated array of elements.

Just like arrays, vectors use <contiguous storage locations > for their elements,

which means that their elements can also be accessed using <offsets> on regular

pointers to its elements, and just as efficiently as in arrays. But unlike arrays,

their <size can change dynamically>, with their storage being handled automatically

by the container.

stod:

Parses str interpreting its content as a floating-point number, which is returned

as a value of type double.

stoi:

Parses str interpreting its content as an integral number of the specified base,

which is returned as an int value.

bool, char, unsigned char, signed char, \_\_int8 1 byte

short, unsigned short, \_\_int16, wchar\_t, \_\_wchar\_t 2 bytes

float, int, unsigned int, long, unsigned long \_\_int32, 4 bytes

double,long double, long long, \_\_int64, 8 bytes

pointer vs reference

A pointer can be re-assigned while reference cannot, and must be assigned at initialization only.

Pointer can be assigned NULL directly, whereas reference cannot.

Pointers can iterate over an array, we can use ++ to go to the next item that a pointer is pointing to.

A pointer is a variable that holds a memory address. A reference has the same memory address as the item it references.

A pointer to a class/struct uses ‘->'(arrow operator) to access it’s members whereas a reference uses a ‘.'(dot operator)

A pointer needs to be dereferenced with \* to access the memory location it points to, whereas a reference can be used directly.

References are usually preferred over pointers whenever we don’t need “reseating”.

Overall, Use references when you can, and pointers when you have to.But if we want to write C code that compiles with both C

and a C++ compiler, you’ll have to restrict yourself to using pointers.

//.at

//.begin

//.clear

//.count

//.empty

//.end

//.erase

//.find

//.insert

//.size

//.swap

array

An array is a series of elements of the <same type> placed in <contiguous memory locations> that can be individually referenced by adding an

index to a <unique identifier>.That means that, for example, five values of type int can be declared as an array without having to declare

5 different variables(each with its own identifier). Instead, using an array, the five int values are stored in contiguous memory locations,

and all five can be accessed using the <same identifier, with the proper index.>

\*/

map

STL Map Internal Implementation :

It’s implemented as a self - balancing red - black tree.Probably the two most common self balancing trees are

red - black tree and AVL trees.To balance the tree after an insertion / update both algorithms use the notion

of rotations where the nodes of the tree are rotated to perform the re - balancing. While in both algorithms

the insert / delete operations are O(log n), in the case of Red - Black tree re - balancing rotation is an O(1)

operation while with AVL this is a O(log n) operation, making the RB tree more efficient in this aspect of the re

- balancing sage and one of the possible reasons that is more commonly used.

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| map | unordered\_map

-------------------------------------------------------- -

Ordering | increasing order | no ordering

| (by default) |

Implementation | Self balancing BST | Hash Table

| <like Red - Black Tree> |

search time | log(n) | O(1)->Average

| O(n)->Worst Case

Insertion time | log(n) + Rebalance | Same as search

Deletion time | log(n) + Rebalance | Same as search

Differences between hash table and STL map

Null Keys : STL Map allows one null key and multiple null values whereas hash table doesn’t allow any null key or value.

Thread synchronization : Map is generally preferred over hash table if thread synchronization is not needed.Hash table is synchronized.

Thread safe : STL Maps are not thread safe whereas Hashmaps are thread safe and can be shared with many threads.

Value Order : In STL map, values are stored in sorted order whereas in hash table values are not stored in sorted order

Searching Time : You can use STL Map or binary tree for smaller data(Although it takes O(log n) time, the number of inputs may be small enough

to make this time negligible) and for large amount of data, hash table is preferred.

Unordered map - hash table

/\* Values are <not stored in sorted order>

Additionally, since hash tables <use the key to find the index that will store the value>,

an insert or lookup can be done in amortised O(1) time(assuming few collisions in the hash table).

In a hash table, one must also handle <potential collisions>.This is often done <by chaining>,

which means to create a linked list of all the values whose keys map to a particular index.

//Implementation of Hash Table : A hash table is traditionally <implemented with an array of linked lists>.

//When we want to insert a key / Value pair, we map the <key to an index> in the array using the hash function.

//The <value> is then inserted into the <linked list at that position

Set

Sets are typically implemented as binary search trees

Sets are containers that store <unique elements> following a <specific order>.

In a set, the value of an element also identifies it(the value is itself the key,

of type T), and each value must be unique.

The value of the elements in a set cannot be modified once in the container (the elements are always const),

but they can be inserted or removed from the container.

Internally, the elements in a set are always sorted following a specific strict

weak ordering criterion indicated by its internal comparison object (of type

Compare).

set containers are generally <slower than unordered\_set containers> to access <individual elements> by their key, but they allow the direct iteration on subsets based on their order.

//.begin

//.clear

//.count

//.empty

//.end

//.erase

//.find

//.insert

//.size

//.swap

// deque

//.back

//.begin

//.end

//.front

//.pop\_back

//.pop\_front

//.clear

//.empty

//.erase

//.insert

//.size

//.swap

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// queue

/\* FIFO queue

queues are a type of container adaptor, specifically designed to operate in a FIFO context(first - in first - out),

where elements are inserted into one end of the container and extracted from the other.

queues are implemented as containers adaptors, which are classes that use an encapsulated object of a specific container class

as its underlying container, providing a specific set of member functions to access its elements.Elements are <pushed into the

"back" >of the specific container and <popped from its "front">.

The underlying container may be one of the standard container class template or some other specifically

designed container class.This underlying container shall support at least the following operations :

//.push\_back

//.front

//.pop\_front

//.back

//.empty

//.size

Double ended queue

deque (usually pronounced like "deck") is an irregular acronym of double-ended queue. Double-ended queues are sequence containers

with dynamic sizes that can be expanded or contracted on both ends (either its front or its back).

Specific libraries may implement deques in different ways, generally as some form of dynamic array. But in any case, they allow for the individual elements to be accessed directly through random access iterators, with storage handled automatically by expanding and contracting the container as needed.

\*/

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// stack

//LIFO stack

//Stacks are a type of container adaptor, specifically designed to operate in a

//LIFO context(last - in first - out), where elements are inserted and extracted only

//from one end of the container.

/\*

stacks are implemented as containers adaptors, which are classes that use an encapsulated

object of a specific container class as its underlying container, providing a specific set

of member functions to access its elements. Elements are pushed/popped from the "back" of

the specific container, which is known as the top of the stack.

\*/

//.pop

//.top

//.push

//.empty

//.size

//.swap

//back

//push\_back

//pop\_back

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Polymorphism:

polymorphism as the ability of a message to be displayed in more than one form. Like a man at a same time is a father, a husband, a employee.

Compile time polymorphism

**Function Overloading**: When there are multiple functions with same name but different parameters then these functions are said to be **overloaded**.

using namespace std;

class Geeks

{

    public:

    // function with 1 int parameter

    void func(int x)

    {

        cout << "value of x is " << x << endl;

    }

    // function with same name but 1 double parameter

    void func(double x)

    {

        cout << "value of x is " << x << endl;

    }

    // function with same name and 2 int parameters

    void func(int x, int y)

    {

        cout << "value of x and y is " << x << ", " << y << endl;

    }

};

int main() {

    Geeks obj1;

    // Which function is called will depend on the parameters passed

    // The first 'func' is called

    obj1.func(7);

    // The second 'func' is called

    obj1.func(9.132);

    // The third 'func' is called

    obj1.func(85,64);

    return 0;

}

value of x is 7

value of x is 9.132

value of x and y is 85, 64

|  |
| --- |
| Operator Overloading  // CPP program to illustrate  #include<iostream>  using namespace std;    class Complex {  private:      int real, imag;  public:      Complex(int r = 0, int i =0)  {real = r;   imag = i;}        // This is automatically called when '+' is used with      // between two Complex objects      Complex operator + (Complex const &obj) {           Complex res;           res.real = real + obj.real;           res.imag = imag + obj.imag;           return res;      }      void print() { cout << real << " + i" << imag << endl; }  };    int main()  {      Complex c1(10, 5), c2(2, 4);      Complex c3 = c1 + c2; // An example call to "operator+"      c3.print();  } |

Output:

12 + i9

Runtime polymorphism

**Pointers to base class**

One of the key features of class inheritance is that a pointer to a derived class is type-compatible with a pointer to its base class. *Polymorphism* is the art of taking advantage of this simple but powerful and versatile feature.

// pointers to base class

#include <iostream>

using namespace std;

class Polygon {

protected:

int width, height;

public:

void set\_values (int a, int b)

{ width=a; height=b; }

};

class Rectangle: public Polygon {

public:

int area()

{ return width\*height; }

};

class Triangle: public Polygon {

public:

int area()

{ return width\*height/2; }

};

int main () {

Rectangle rect;

Triangle trgl;

Polygon \* ppoly1 = &rect;

Polygon \* ppoly2 = &trgl;

ppoly1->set\_values (4,5);

ppoly2->set\_values (4,5);

cout << rect.area() << '\n';

cout << trgl.area() << '\n';

return 0;

}

**Virtual members**

A virtual member is a member function that can be redefined in a derived class, while preserving its calling properties through references. The syntax for a function to become virtual is to precede its declaration with the virtual keyword:

**Abstract base classes**

Abstract base classes are something very similar to the Polygon class in the previous example. They are classes that can only be used as base classes, and thus are allowed to have virtual member functions without definition (known as pure virtual functions). The syntax is to replace their definition by =0 (an equal sign and a zero):

// pure virtual members can be called

// from the abstract base class

#include <iostream>

using namespace std;

class Polygon {

protected:

int width, height;

public:

void set\_values (int a, int b)

{ width=a; height=b; }

virtual int area() =0;

void printarea()

{ cout << this->area() << '\n'; }

};

class Rectangle: public Polygon {

public:

int area (void)

{ return (width \* height); }

};

class Triangle: public Polygon {

public:

int area (void)

{ return (width \* height / 2); }

};

int main () {

Rectangle rect;

Triangle trgl;

Polygon \* ppoly1 = &rect;

Polygon \* ppoly2 = &trgl;

ppoly1->set\_values (4,5);

ppoly2->set\_values (4,5);

ppoly1->printarea();

ppoly2->printarea();

return 0;

}

Virtual

//Virtual members/virtual function/ virtual method

A virtual member

is a member function that can be redefined in a derived class,

//The member function area has been declared as virtual in the base class because it is later redefined ssset

//in each of the derived classes.Non - virtual members can also be redefined in derived classes,

//non - virtual members of derived classes cannot be accessed through a reference of the base class

// : i.e., if virtual is removed from the declaration of area in the example above,

// all three calls to area would return zero, because in all cases, the version of the base class would

// have been called instead.

Virtual function:

In [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B), and [Object Pascal](https://en.wikipedia.org/wiki/Object_Pascal), a **virtual function**or **virtual method** is an inheritable and [overridable](https://en.wikipedia.org/wiki/Method_overriding_(programming)) [function](https://en.wikipedia.org/wiki/Function_(computer_science)) or [method](https://en.wikipedia.org/wiki/Method_(computer_science)) for which [dynamic dispatch](https://en.wikipedia.org/wiki/Dynamic_dispatch)is facilitated. This concept is an important part of the (runtime) [polymorphism](https://en.wikipedia.org/wiki/Polymorphism_(computer_science)) portion of [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) (OOP). In short, a virtual function defines a target function to be executed, but the target might not be known at compile time.

Pure virtual:

A **pure virtual function** or **pure virtual method** is a virtual function that is required to be implemented by a derived class if the derived class is not [abstract](https://en.wikipedia.org/wiki/Abstract_type). Classes containing pure virtual methods are termed "abstract" and they cannot be instantiated directly.

also called abstract class. Pure virtual cannot new object.

Son must define a virtual function.

Why use pure virtual?

<To be sure the subclass implements>

Function overloading

is a feature in C++ where two or more functions can have the same name but different parameters.

Function overloading can be considered as an example of polymorphism feature

in C++.

#include <iostream>

using namespace std;

void print(int i) {

  cout << " Here is int " << i << endl;

}

void print(double  f) {

  cout << " Here is float " << f << endl;

}

void print(char const \*c) {

  cout << " Here is char\* " << c << endl;

}

int main() {

  print(10);

  print(10.10);

  print("ten");

  return 0;

}

**Function Overriding (achieved at run time)**

It is the redefinition of base class function in its derived class with same signature i.e return type and parameters.

|  |
| --- |
| // Function Overriding  #include<iostream>  using namespace std;    class BaseClass  {  public:      virtual void Display()      {          cout << "\nThis is Display() method"                  " of BaseClass";      }      void Show()      {          cout << "\nThis is Show() method "                 "of BaseClass";      }  };    class DerivedClass : public BaseClass  {  public:      // Overriding method - new working of      // base class's display method      void Display()      {          cout << "\nThis is Display() method"                 " of DerivedClass";      }  };    // Driver code  int main()  {      DerivedClass dr;      BaseClass &bs = dr;      bs.Display();      dr.Show();  } |
|  |

Output:

This is Display() method of DerivedClass

This is Show() method of BaseClass

**Function Overloading VS Function Overriding**

1. **Inheritance:** Overriding of functions occurs when one class is inherited from another class. Overloading can occur without inheritance.
2. **Function Signature:** Overloaded functions must differ in function signature ie either number of parameters or type of parameters should differ. In overriding, function signatures must be same.
3. **Scope of functions:** Overridden functions are in different scopes; whereas overloaded functions are in same scope.
4. **Behavior of functions:**Overriding is needed when derived class function has to do some added or different job than the base class function. Overloading is used to have same name functions which behave differently depending upon parameters passed to them.

Data Structure

1. Describe the implementation of array, linked-list and binary tree. (Describe how the memory is allocated. how the nodes are connected).

Binary tree: block of memory

2. Describe the adv and disadv of array and linked-list

Array: linked list

Fixed size Dynamic size(resize O(1))

Inefficient insertions and deletions efficient insertions and deletions

Random access no random access

Sequential access is fast(traverse) sequential access is slow(O(n)

Access the previous element easily.

Binary search(sorted array) no binary search

memory

contiguous memory location not in contiguous memory location

May result in much memory waste no waste of memory

Indexing(get)

O(1) O(n)

Insert at the beginning

O(n) O(1)

Insert at the end

O(1) O(1) if the list has tail reference

O(n) if the list has no tail reference

Insert at the middle (need to move all the index, only need to delete the node)

O(n) O(1)

Delete at the beginning

O(n) O(1)

Delete at the end

O(1) O(n)

delete at the middle (need to move all the index, only need to delete the node)

O(n): O(n)

O(1) access followed by O(n) shift O(n)search, followed by O(1) delete

Push and pop at end

O(1) O(1)

Travers

O(n) (linear search) O(n)

O(logn) (binaray search(

Constructor:

is automatically called whenever a new object of this class is created, allowing the class to initialize member variables or allocate storage.

Like any other function, a constructor can also be overloaded with different versions taking different parameters: with a different number of parameters and/or parameters of different types. The compiler will automatically call the one whose parameters match the arguments:

Overloading constructors:

Rectangle::Rectangle () {

Rectangle::Rectangle (int a, int b) {

Destructor

A destructor is a special [member function](https://en.cppreference.com/w/cpp/language/member_functions) that is called when the [lifetime of an object](https://en.cppreference.com/w/cpp/language/lifetime) ends. The purpose of the destructor is to free the resources that the object may have acquired during its lifetime.

Memory leaks

occur when new memory is allocated dynamically and never deallocated.

Template:

Template is simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don’t need to write same code for different data types. For example a software company may need sort() for different data types. Rather than writing and maintaining the multiple codes, we can write one sort() and pass data type as a parameter.

Compiler:

Computers understand only one language and that language consists of sets of instructions made of ones and zeros. This computer language is appropriately called *machine language*.

The compilation of a C++ program involves three steps:

1. Preprocessing: the preprocessor takes a C++ source code file and deals with the #includes, #defines and other preprocessor directives. The output of this step is a "pure" C++ file without pre-processor directives.
2. Compilation: the compiler takes the pre-processor's output and produces an object file from it.
3. Linking: the linker takes the object files produced by the compiler and produces either a library or an executable file.

Inheritance:

The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important feature of Object Oriented Programming.  
**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.  
**Super Class:**The class whose properties are inherited by sub class is called Base Class or Super class.

access specifier

**Public**: All the class members declared under public will be available to everyone.

**Private**: The class members declared as **private** can be accessed only by the functions inside the class.

**Protected**: Protected access modifier is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass(derived class) of that class.

In-order: left->node->right

if ( root != nullptr )

{

displayInOrder( root->left );

cout << root->value << endl;

displayInOrder(root->right);

}

if ( root != nullptr )

{

cout << root->value << endl;

displayPreOrder( root->left );

displayPreOrder( root->right );

}

if ( root != nullptr )

{

displayPostOrder( root->left );

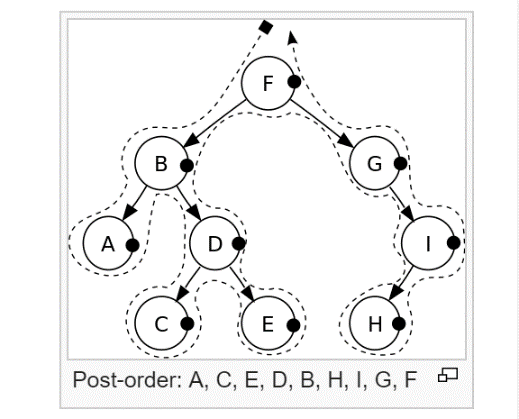
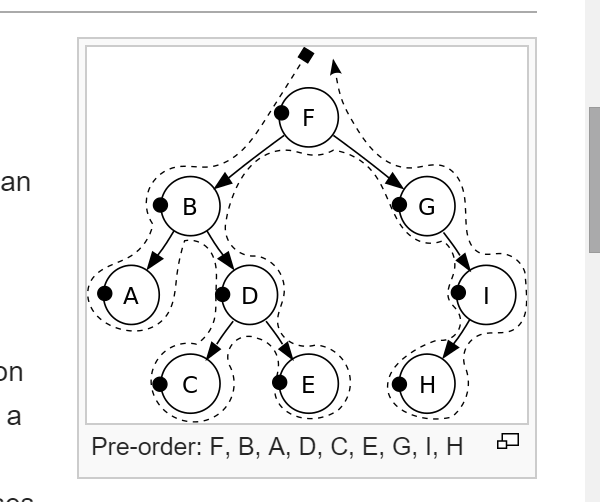
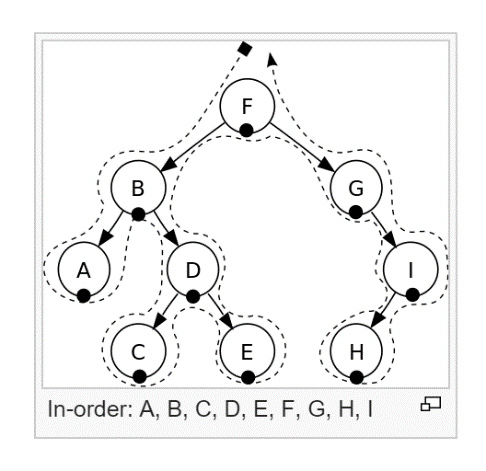
displayPostOrder( root->right );

cout << root->value << endl;

}

Pre-order: node->left->right

Post-order: left->right->node



Data structure /struct

 data structure is a group of data elements grouped together under one name. These data elements, known as *members*, can have different types and different lengths. Data structures can be declared in C++ using the following syntax:

Bit calculate

1 bit = 0 or 1 (b)

8 bits = 1 byte (B)

1 thousand bytes = kilobyte (KB) 1,000

1 million bytes = megabyte (MB) 1,000,000

1 billion bytes = gigabyte (GB) 1,000,000,000

1 trillion = terabyte (GB) 1,000,000,000,000

1 bit = 0 or 1 (b)

Nibble = 4 bits

1 byte/Octet = 8 bytes (B)

1 Kilobyte (KB) = 1024 bytes

1 Megabyte (MB) = 1024(KB)

1 Gigabyte (GB) = 1024 (MB)

1 Terabyte (GB) = 1024 (GB)

1 Exabyte (GB) = 1024 (TB)

1 Zettabyte (GB) = 1024 (EB)

1 Yottabyte (GB) = 1024 (ZB)

3. Know the implementation of BFS and DFS on tree. Need Pseudo C++ code here.

4. How to DFS a graph? which means you may have cycles in the graphs?

5. How does hash table/map work?

6. What is set?

C++

1. What is constructor and destructor, when will they get called?

2. What is memory leak?

3. What is the size of byte? kilobyte? megabyte?

4. What is function overload?

5. What is virtual function? What is function override? (This is realted to "Multimorphism" 多种形态， 简称多态)

6. What is template?

7. What is multiple threading? What happens when 10 threads want to operate the same memory location? (Hint: read after read, write after read, write after write)

8. What is compile? What is link?

9. What is access specifier? public protected private?

10. What is bit operation and/or/xor? How to set a bit? clear a bit?

11. What is pointer/address? Pointer to a pointer?

Maths:

1. What is definition of |a| ? absolute value? |-1| = ? |3-6] = ?

Algorithm:

1. Inverse a linked list.

2. How to hash a string?

3. If you have a 2d array V. size is NxN. the element in the V is either 0 or 1. for example:

V = [0, 0, 0, 1, 0, 0]

[0, 1, 0, 1, 1, 0]

[0, 1, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 1]

[0, 0, 0, 0, 1, 1]

[1, 1, 0, 0, 1, 0]

Question: if there is V[i][j] = 1. and V[i][j]'s up or down or left or right neighbor V' is 1. Then we say V and V' are connected in one group. In this example we see 4 connected groups. Please find out how many groups in the 2d array. Hint: DFS.

Problem solving:

1. Write a pseudo code to handle the behavior of an elevator class. You decide the methods and memebrs by yourself.

2. Write a pseudo code to handle the behavior of a car class. You decide methods and members by yourself.

1. Which of the following is a correct comment?

/\* \*/

Top of Form

**2.Which of the following is true?**

~~if(1)?~~



~~if(66)~~



i~~f(.1)?~~



~~if( -1)?~~



All of the above\*



Top of Form

**3.Evaluate! (1 && ! (0 || 1)).**

True\*



~~False?~~



~~Unevaluatable?~~



**4.Which is not a loop structure?**

For



Do while



While



Repeat Until\*



Top of Form

**5.When does the code block following while(x<100) execute?**

When x is less than one hundred?\*



~~When x is greater than one hundred?~~



~~When x is equal to one hundred~~



~~While it wishes~~



**6.What character ends all strings?**

'.'



' '



'\0' \*



'\n'



**x7.Which of the following reads in a string named x with one hundred characters?**

fgets(x, 101, stdin);\*



~~fgets(x, 100, stdin);?~~



~~readline(x, 100, '\n');?~~



~~read(x);?~~



Top of Form

**8.Which of the following is the proper keyword to allocate memory in C?**

~~new~~



malloc()\*



~~create~~



~~value?~~



**9.Which of the following accesses a variable in structure \*b?**

b->var;\*



~~b.var;~~



~~b-var;~~



~~b>var;?~~



**10.Consider the following program fragment.   
d = 0;  
for(i = 1; i < 31; ++i)   
for(j = 1; j < 31; ++j)   
for(k = 1; k < 31; + +k)  
if ( ( (i + j + k) % 3) == 0)  
d = d+1;  
printf("%d",d);  
The output will be**

9000\*



~~27000?~~



~~3000~~



~~none of the above~~



Top of Form

**x11.Are the following declarations same?  
  
char far \*far \*scr;  
char far far\*\* scr;**

Yes



No \*



**12.Functions cannot return a floating point number**

Yes



No \*



:00:06:54

Top of Form

**x13.The statement in the previous question can never print**

~~one~~



two\*



~~three~~



~~four~~



**14.What will be the output of the program ?  
#include<stdio.h>  
#include<string.h>  
int main()  
{  
char str1[5], str2[5];  
int i;  
gets(str1);  
gets(str2);  
i = strcmp(str1, str2);  
printf("%d\n", i);  
return 0;  
}**

Unpredictable integer value\*



~~0~~



~~-1?~~



~~Error~~



**x15.If the two statements  
\* p v = 0;  
printf ("%d %d", \*pv, v);  
are appended to the previous program fragment then the output will be**

~~0 3?~~



0 0\*



~~unpredictable?~~



* ~~none of the above?~~

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hint. Don't forget the all of the strings has a hidden char = 0 or '\0' and the end.

1. Suppose the function "fun" has two inputs string s ( char \*) and char c, the function will search in s to find if there is a char having same value as c. if found, insert c again after this char.

If input s = "bbaacda" c = 'a' . then result s = "bbaaaacdaa"

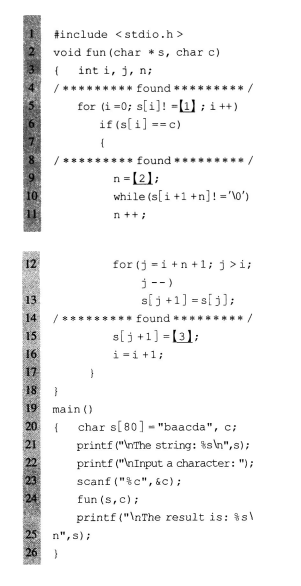
try to find what to put in the blank [1] [2] and [3]

[1]: ‘\0’

[2]: ‘0’

[3]: ‘c’

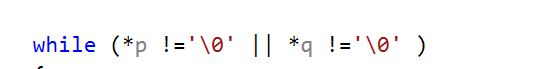
any memory that allocated to you is random. if you don't do initialize. anything is random.

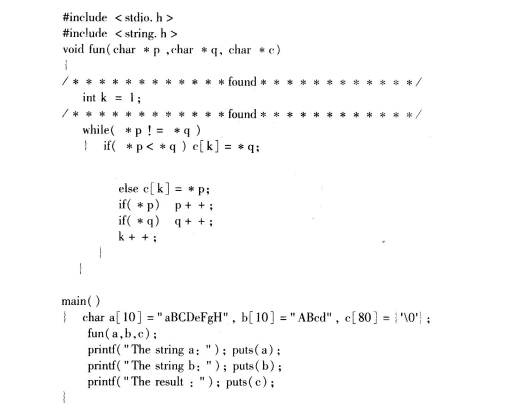


2. The function "fun" will try to compare the two strings in p and q from the first position to the last. The comparing criteria is to find the char which has larger ASCII value. It will put the comparision result in string c. If p and q don't have the same length, then only compare the part with same length, and put the rest of the longer string directly to c.

If p = "aBCDeFgH" q = "Abcd" . then c = "abcdeFgH" because 'a' > 'A', 'b' > 'B', 'c' > 'C', 'd' > 'D'

There are two line of code have bugs. please find them and correct them.

1   
  
C:\Users\Yu\Documents\Tencent Files\651412330\Image\C2C\`MILZW444]ZCOU{A37{KGA3.png  
  
  
  




char a[10] = "aBC", b[10] = "ABcdaaaaa", c[80] = {'a', '\0' };

fun(a, b, c);

printf("The string a: "); puts(a);

printf("The string b: "); puts(b);

printf("The string c: "); puts(c);

void fun(char \*p, char \*q, char \*c)

{

int k = 0;

while (\*p !='\0' || \*q !='\0' )

{

if (\*p < \*q)

{

c[k] = \*q;

}

else

{

c[k] = \*p;

}

k++;

if (\*p)

{

p++;

}

if (\*q)

{

q++;

}

}

}

// check if a positive integer n can be represented by sum of some consecutive integers.

// if n = 100.

// then output 100 = 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16

// 100 = 18 + 19 + 20 + 21 + 22

// Two error in the code.

#include <stdio.h>

void fun(int n)

{

int j, b, c, m, flag = 0;

for (b = 1; b <= n / 2; b++) {

n = m; // error m = n

c = b;

while (m != 0 && m >= c) {

m = m - c;

c++;

}

if (m != 0) { // error m == 0

printf("%d = ", n);

for (j = b; j < c - 1; j++)

printf("%d + ", j);

printf("%d\n", j);

flag = 1;

}

}

if (flag == 0) {

printf("Couldn't find.");

}

}

int main()

{

int n;

printf("input: ");

scanf\_s("%d", &n);

fun(n);

}

class design practice. write classes in c++

Create classes for Truck, Car and SUV. Your design must meet the following requirments:

1. Those three wil have shared attibutes: 1. horse power(int). 2. seat number(int). 3. MPG(int). 4. Make(stirng). 5. Model(stirng). 6. Vehicle Type(enum, google how to use enum in c++) These shared attributes must be put in the parent class Vehicle. Make sure you understand the meaning of the word in attribute. If you don't know, you can search google, i.e., what is the MPG of a vehicle?

2. These attributes CANNOT be accessed directly from outside of the class (trying to use the attritbutes will cause a build error). Must use member function GetAttributeName or SetAttributeName.

3. Each child class has it's own attributes:. For truck, it has its own member Towing Capacity (float). For car, it has its own member Car Type (it should be an enum of Coupe and Sedan, if you don't know how to use enum, you can google). For SUV, it has its own member AWD (bool, check the meaning of AWD if you don't now). And their own get and set. These members are also not allowed to be accessed from outside, need to use Get and Set to access.

4. Set the Vehicle Type (enum) in the constructor.

5. If an object of Truck, Car and SUV, is pointed by a Vehicle pointer, you should be able to call the member function "GetVehicleType" of Vehicle to get the type of the object. It should be an enum of Truck, Car or SUV. Here there is NO need to use virtual function.

6. If an object of Truck, Car and SUV, is pointed by a Vehicle pointer, you should be able to call the member function "GetMonthlyCost" of vehicle to get different monthly cost for the truck, car and suv. For truck, the monthly cost is MPG \* Tow Capacity / 100; for car, the montly cost is MPG \* 30 for sedan, MPG \* 35 for coupe; for SUV the monthly cost is MPG \* 40 for not AWD, MPG \* 45 for AWD. This one use virtual function.

7. You can build it without any problems.

8. You can run the following main function and get correct results. And you can get build failure after you uncomment line "std::cout << v->mpg << std::endl; "

int main()

{

Car car1, car2;

Truck truck1, truck2;

SUV suv1, suv2;

Vehicle \*v = nullptr;

car1.SetMPG(30);

car1.SetCarType(CAR\_TYPE\_SEDAN); // this will work after you add the enum

car2.SetMPG(30);

car2.SetCarType(CAR\_TYPE\_COUPE); // this will work after you add the enum

truck1.SetMPG(20);

truck1.SetTowingCapacity(6000);

truck2.SetMPG(20);

truck2.SetTowingCapacity(5000);

suv1.SetMPG(25);

suv1.SetAWD(true);

suv2.SetMPG(25);

suv2.SetAWD(false);

v = &car1;

// this line SHOULD cause a BUILD FAILE.

// After you cause the build fail, you can comment it out.

// if you cannot cause build fail. Your code is wrong.

// std::cout << v->mpg << std::endl;

// This will work after you add the enum

if (v->GetVehicleType() == VEHICLE\_TYPE\_CAR) {

std::cout << "test 1 passed\n";

}

else {

std::cout << "test 1 failed\n";

}

v = &car1;

if (v->GetMonthlyCost() == 900) {

std::cout << "test 2 passed\n";

}

else {

std::cout << "test 2 failed\n";

}

v = &truck2;

if (v->GetMonthlyCost() == 1000) {

std::cout << "test 3 passed\n";

}

else {

std::cout << "test 3 failed\n";

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

she talked about her company, what kind of positions is opening.(could write down on the note)

which type I prefer? Only coding only or coding with consulting?(I need to ask more detail about each type of job)

michael

Difference between passing by reference and by value

<https://courses.washington.edu/css342/zander/css332/passby.html>

passing by address:  a copy of the address of the actual parameter is stored. Use pass by reference when you are changing the parameter passed in by the client program.

passing by value:  means you are making a copy in memory of the actual parameter's value that is passed in, a copy of the contents of the actual parameter. Use pass by value when when you are only "using" the parameter for some computation, not changing it for the client program. .

the types of loops. (while, do while, for, nested loop)

If people do not know about inheritance, how do you explain the concept to them.

<https://www.programiz.com/cpp-programming/inheritance>

* + 1. Object-oriented programming in C++;
    2. to create a new [class](https://www.programiz.com/cpp-programming/object-class) (derived class) from an existing class(base class).
    3. The derived class inherits all the features from the base class and can have additional features of its own.
    4. It is important to remember: When working with inheritance, each derived class should satisfy the condition whether it "is a" base class or not. In the example above, Maths teacher is a Person, Footballer is a Person. You cannot have: Businessman is a Business.

<http://bigocheatsheet.com/>

C:\Users\Yu\AppData\Local\Temp\%W@GJ$ACOF(TYDYECOKVDYB.pnghttps://projecteuler.net/archives // sample questions.

Interview question

**memset**( pointer, size, valueperbyte )

就是把从pointer开始 到size这么大的memory, 每一个byte都设成 valueperbyte的值

**memcpy**( b, a, sizeof( int ) \* 100 )  
就是把一块memory 直接复制到另一块memory, size自己定 超快, 比for 快很多

char source[] = "once upon a midnight dreary...", dest[4];

std::memcpy(dest, source, sizeof dest); // dest: once // only the size of dest. 4 byte.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

int n;   
if (n == 10) // 第一种判断方式   
if (10 == n) // 第二种判断方式  
哪种对, 如果都对 哪种更好

the second one is better.

If you miss = , the second one will not compile.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Void test1()   
{   
 char string[10];   
 char \*str1="0123456789";  
 strcpy(string, str1);   
}

What wrong with this function?

Do not have enough memory, because will have a empty char at the end.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Char a = -1, is it ok, if ok, what is the memory. // if it is char, it is 1byte ff

0000 0001->1111 1110+1->1111 1111 ->FF

6 is 4 byte.

FFFF FFFF

FFFF FFFE

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Depth First Tree Traversal是什么 Breadth First Tree Traversal

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

threading里面的 dead lock是什么

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

implement一个除法, 不用/ 除号, 用列竖式的原理

Bit manipulation. Shift.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

有个bool的array 长度64, 能把这个bool array存在两个int里面吗?

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

怎么找一个single linked list的中点?

第一个pointer走一步第二个pointer走两步

走到头了 第一个pointer 就是中点

if ( second->next != null )

{

second = second->next;

if ( second->next != null )

{

second = second->next;

first = first->next;

}

else

{

the midPoint is the firstP->value and the firstP->next->value;

return;

}

}

else

{

the midPoint is the firstP;

return;

}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

new 和 malloc 区别

new 是一个pointer 没有size， malloc 要有size 吧？

new 就是 allocate mamory的同时还call constructor, 但是 malloc只 allocate memory

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

最后一个问题 如果有个 parent class 有个 child class, 那么destructor 为什么要是virtual的

When you create, it is at son class. So need to delete son destructor first.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

等差数列 1+2+3+….. 100

100+99+98……1

(1+100)\*100/2

(1+n)\*n/2

(N+n^2)/2

1+3+5+……+99.

1+2\*m = 99

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

C:

no class in c(no constructor).

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Instantiate: create a new object called instantiate.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Class: default access specifier is private.

Struct: default access specifier is public.

The size of empty class is 1 byte.

The size of class equal to the total private member together (not the public).

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

int \*p = 0x90, the next pointer address will be 0x94

byte \*p= 0x90, the next pointer address will be 0x91

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Always remember if you have if then need to have else.

Always check the situation. Before you do you other process. Like if (a <0){ return false}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

**Why use using namespace std?**

All the variable or functions included in namespace std, we will use it.

If you define the same name variable or function in your class. The compiler do not know which one you are using.

So the namespace std avoid duplicate name.

Namespace std control all the include libarary: queue, stack, list…ect.

#include<queue>

Template<class T>

Class queue

{

T a;

}

Int main()

{

Queue<int > a;

Std::queue<int>a;

Return 0;

}

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

给你出一个题 已知 rand() 可以generate a number from 0 to 65536, please use this function to generate randome number in { 10, 20, 30, 40, 50 }

 for ( int i = 0; i < 10; i++ )  
    {  
        a = rand() % 5;  
        a = ( a + 1 ) \* 10;  
        std::cout << a << std::endl;  
    }

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

再给你出个题 a \* 67 这个式子 不用乘法 不用loop 你能换一种写法吗

a<<6 +a<<1+1;

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

hashtable 原理, hashing是什么, 怎么就直接能在array里找到一个key对应的位置

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

3. depth first traverse tree, 和 breath first traverse tree

Depth: preorder, inorder, and postorder.

Breath: step for each lever. Cout<<node, and the put it next to the queue. And then pop out the node. End the queue is empty.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

4. pre order in order post order traverse tree

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

5. smartpointer

auto\_ptr<student> aptr( new student( "Bob", "Smith", "CIS29", "4.0" ) );

unique\_ptr<student> uptrA( new student( "Bob", "Smith", "CIS29", "4.0" ) );

unique\_ptr is autoptr’s replacement

//unique\_ptr<student> uptrB = uptrA; // compile error - creates copy

unique\_ptr<student> uptrB = move( uptrA );

unique\_ptr<student> uptrC = static\_cast<unique\_ptr<student>&&>( uptrA );

shared\_ptr<student> sptrA( new student( "Alan", "Johnson", "CIS29", "4.0" ) );

shared\_ptr<student> sptrB = sptrA;

shared\_ptr<student> sptrC = sptrB;

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

怎么知道int a 能否被int b整除

a%b==0

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

在一个字符串中找到第一个只出现一次的字符。如输入abaccdeff，则输出b。

S[char]++

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

输入一个整数，求该整数的二进制表达中有多少个1。

例如输入10，由于其二进制表示为1010，有两个1，因此输出2。

Check until 32 bit

Use check function.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

在从1到n的正数中1出现的次数

题目：输入一个整数n，求从1到n这n个整数的十进制表示中1出现的次数。

例如输入12，从1到12这些整数中包含1 的数字有1，10，11和12，1一共出现了5次。

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

什么是deadlock

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

From string to char.

String mystring=”fkdjfkd”;

char \* p = mystring.c\_str;

defind a pointer to an array.

char \*p="abcdefg";

string a = p; //output: abcdefg

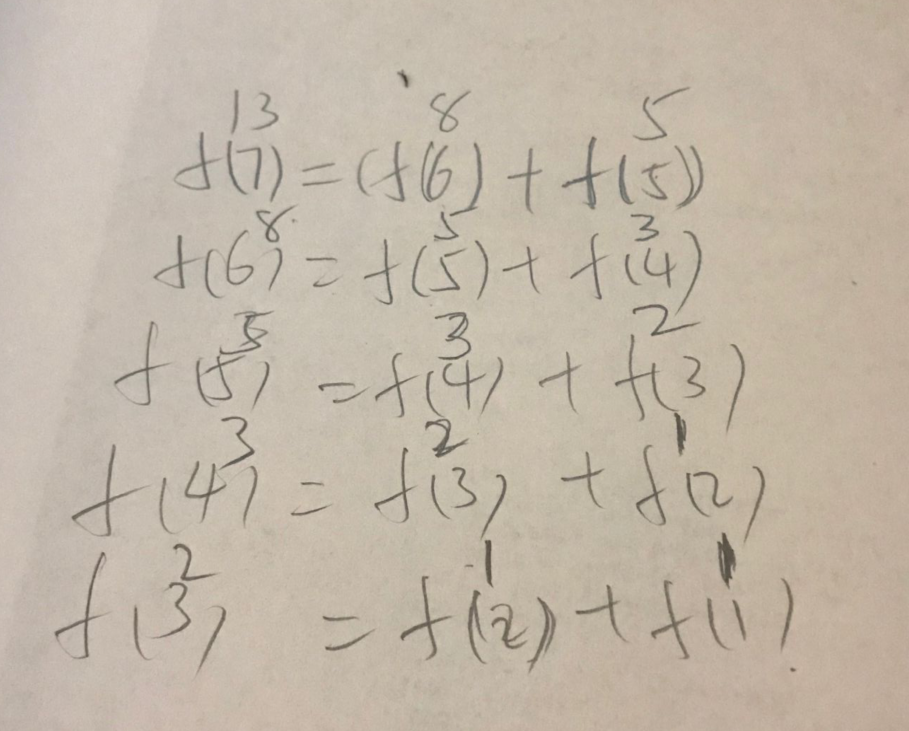
int to string

to\_string(14);

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

Peek equal to top

4/11/



2. give me a function int f(int n). which will return the nth element in the array:      
C:\Users\Yu\Documents\Tencent Files\651412330\Image\C2C\5M233LW4UR~1}55ZO_E[M{Q.png  
  
hint.  F(n) = F(n-1) + F(n-2)

1. do a recursion function int sum( int n )  
if input is n, you reutrn 1+2+... \_+ n

3/30

class design practice. write classes in c++

Create classes for Truck, Car and SUV. Your design must meet the following requirments:

1. Those three wil have shared attibutes: 1. horse power(int). 2. seat number(int). 3. MPG(int). 4. Make(stirng). 5. Model(stirng). 6. Vehicle Type(enum, google how to use enum in c++) These shared attributes must be put in the parent class Vehicle. Make sure you understand the meaning of the word in attribute. If you don't know, you can search google, i.e., what is the MPG of a vehicle?

2. These attributes CANNOT be accessed directly from outside of the class (trying to use the attritbutes will cause a build error). Must use member function GetAttributeName or SetAttributeName.

3. Each child class has it's own attributes:. For truck, it has its own member Towing Capacity (float). For car, it has its own member Car Type (it should be an enum of Coupe and Sedan, if you don't know how to use enum, you can google). For SUV, it has its own member AWD (bool, check the meaning of AWD if you don't now). And their own get and set. These members are also not allowed to be accessed from outside, need to use Get and Set to access.

4. Set the Vehicle Type (enum) in the constructor.

5. If an object of Truck, Car and SUV, is pointed by a Vehicle pointer, you should be able to call the member function "GetVehicleType" of Vehicle to get the type of the object. It should be an enum of Truck, Car or SUV. Here there is NO need to use virtual function.

6. If an object of Truck, Car and SUV, is pointed by a Vehicle pointer, you should be able to call the member function "GetMonthlyCost" of vehicle to get different monthly cost for the truck, car and suv. For truck, the monthly cost is MPG \* Tow Capacity / 100; for car, the montly cost is MPG \* 30 for sedan, MPG \* 35 for coupe; for SUV the monthly cost is MPG \* 40 for not AWD, MPG \* 45 for AWD. This one use virtual function.

7. You can build it without any problems.

8. You can run the following main function and get correct results. And you can get build failure after you uncomment line "std::cout << v->mpg << std::endl; "

int main()

{

Car car1, car2;

Truck truck1, truck2;

SUV suv1, suv2;

Vehicle \*v = nullptr;

car1.SetMPG(30);

car1.SetCarType(CAR\_TYPE\_SEDAN); // this will work after you add the enum

car2.SetMPG(30);

car2.SetCarType(CAR\_TYPE\_COUPE); // this will work after you add the enum

truck1.SetMPG(20);

truck1.SetTowingCapacity(6000);

truck2.SetMPG(20);

truck2.SetTowingCapacity(5000);

suv1.SetMPG(25);

suv1.SetAWD(true);

suv2.SetMPG(25);

suv2.SetAWD(false);

v = &car1;

// this line SHOULD cause a BUILD FAILE.

// After you cause the build fail, you can comment it out.

// if you cannot cause build fail. Your code is wrong.

// std::cout << v->mpg << std::endl;

// This will work after you add the enum

if (v->GetVehicleType() == VEHICLE\_TYPE\_CAR) {

std::cout << "test 1 passed\n";

}

else {

std::cout << "test 1 failed\n";

}

v = &car1;

if (v->GetMonthlyCost() == 900) {

std::cout << "test 2 passed\n";

}

else {

std::cout << "test 2 failed\n";

}

v = &truck2;

if (v->GetMonthlyCost() == 1000) {

std::cout << "test 3 passed\n";

}

else {

std::cout << "test 3 failed\n";

}

}

3/29

// check if a positive integer n can be represented by sum of some consecutive integers.

// if n = 100.

// then output 100 = 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16

// 100 = 18 + 19 + 20 + 21 + 22

// Two error in the code.

#include <stdio.h>

void fun(int n)

{

int j, b, c, m, flag = 0;

for (b = 1; b <= n / 2; b++) {

n = m; // error m = n

c = b;

while (m != 0 && m >= c) {

m = m - c;

c++;

}

if (m != 0) { // error m == 0

printf("%d = ", n);

for (j = b; j < c - 1; j++)

printf("%d + ", j);

printf("%d\n", j);

flag = 1;

}

}

if (flag == 0) {

printf("Couldn't find.");

}

}

int main()

{

int n;

printf("input: ");

scanf\_s("%d", &n);

fun(n);

}

3/27

Hint. Don't forget the all of the strings has a hidden char = 0 or '\0' and the end.

1. Suppose the function "fun" has two inputs string s ( char \*) and char c, the function will search in s to find if there is a char having same value as c. if found, insert c again after this char.

If input s = "bbaacda" c = 'a' . then result s = "bbaaaacdaa"

try to find what to put in the blank [1] [2] and [3]

[1]: ‘\0’

[2]: ‘0’

[3]: ‘c’

any memory that allocated to you is random. if you don't do initialize. anything is random.

1. 1MB can hold how many integers?

1MB= 1024 KB

1KB = 1024 byte

1024\*1024/4= 1048576/4=262144

2. What is high level language and low-level language?

Human languages: Chinese, English

High level language: C++, java

Low level language: assembly is the normal language

Machine language: binary.

3. Multithreading, Mutex & Semaphore

Multithreading: share memory in the process (data, code, heap), each threading has own stack,

register,

mutex: have one lock

semaphore: have more than one lock.

Mutex and semaphore work between threads or process.   
4. Virtual Memory (Paging)

Each process has 4G memory, have many partitions(4K). Each partition address on virtual memory and physical memory saved to the page table.

5. What is heap, stack in memory?

Heap: dynamic memory, shared memory.

Stack: static memory, for each thread.

6. How to allocate array? How to allocate linked list?

Array: Continues

Linked list: could link to different location.

7. How to delete the entire linked list?

Delete the value in the linked list and unlink it.

8. What is "buffer"?

Like input, waiting for something.

Wrong. Buffer is like a block of contiguous memory to hold data. Input is just an specific example. Anything like:

char array[500];

int arry1[1000];

are buffers.

9. What is Object Oriented Programming?

Everything is class. Constructor

10. What is inheritance?

Son has all the features of the father.

Call son’s construction, also will call the father’s constructor.

11. What is polymorphism? (Function overloading and overriding?)

Some type of things. Dog: jingba or heibei .

You need to be more specific. like different between function overloading and overriding. Will test you later.

12. What is Standard Template Library (STL) in C++?

Download other people’s library, when you use it , just include the library and use their functions.

Wrong. STL are standard libraries like vector, queue, list, stack what are provided by C++. it comes with template. when you declare, you do:

std::vector<int> a;

std::queue<float> b;

different template holds differnt type of data.

1. Which of the following is a correct comment?

/\* \*/

Top of Form

**2.Which of the following is true?**

~~if(1)?~~



~~if(66)~~



i~~f(.1)?~~



~~if( -1)?~~



All of the above\*



Top of Form

**3.Evaluate! (1 && ! (0 || 1)).**

True\*



~~False?~~



~~Unevaluatable?~~



**4.Which is not a loop structure?**

For



Do while



While



Repeat Until\*



Top of Form

**5.When does the code block following while(x<100) execute?**

When x is less than one hundred?\*



~~When x is greater than one hundred?~~



~~When x is equal to one hundred~~



~~While it wishes~~



**6.What character ends all strings?**

'.'



' '



'\0' \*



'\n'



**x7.Which of the following reads in a string named x with one hundred characters?**

fgets(x, 101, stdin);\*



~~fgets(x, 100, stdin);?~~



~~readline(x, 100, '\n');?~~



~~read(x);?~~



Top of Form

**8.Which of the following is the proper keyword to allocate memory in C?**

~~new~~



malloc()\*



~~create~~



~~value?~~



**9.Which of the following accesses a variable in structure \*b?**

b->var;\*



~~b.var;~~



~~b-var;~~



~~b>var;?~~



**10.Consider the following program fragment.   
d = 0;  
for(i = 1; i < 31; ++i)   
for(j = 1; j < 31; ++j)   
for(k = 1; k < 31; + +k)  
if ( ( (i + j + k) % 3) == 0)  
d = d+1;  
printf("%d",d);  
The output will be**

9000\*



~~27000?~~



~~3000~~



~~none of the above~~



Top of Form

**x11.Are the following declarations same?  
  
char far \*far \*scr;  
char far far\*\* scr;**

Yes



No \*



**12.Functions cannot return a floating point number**

Yes



No \*



:00:06:54

Top of Form

**x13.The statement in the previous question can never print**

~~one~~



two\*



~~three~~



~~four~~



**14.What will be the output of the program ?  
#include<stdio.h>  
#include<string.h>  
int main()  
{  
char str1[5], str2[5];  
int i;  
gets(str1);  
gets(str2);  
i = strcmp(str1, str2);  
printf("%d\n", i);  
return 0;  
}**

Unpredictable integer value\*



~~0~~



~~-1?~~



~~Error~~



**x15.If the two statements  
\* p v = 0;  
printf ("%d %d", \*pv, v);  
are appended to the previous program fragment then the output will be**

~~0 3?~~



0 0\*



~~unpredictable?~~



~~none of the above?~~



Bottom of Form

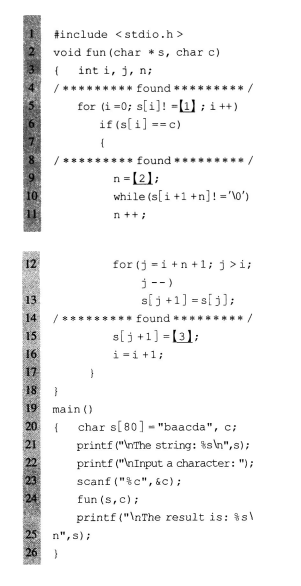
Bottom of Form

Bottom of Form

Bottom of Form

Bottom of Form

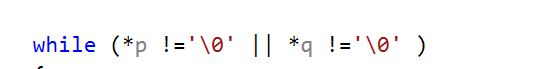
Bottom of Form

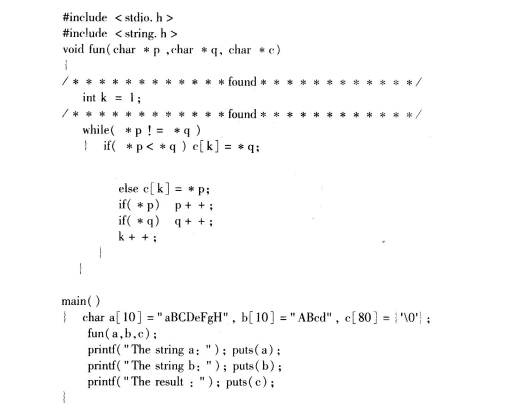


2. The function "fun" will try to compare the two strings in p and q from the first position to the last. The comparing criteria is to find the char which has larger ASCII value. It will put the comparision result in string c. If p and q don't have the same length, then only compare the part with same length, and put the rest of the longer string directly to c.

If p = "aBCDeFgH" q = "Abcd" . then c = "abcdeFgH" because 'a' > 'A', 'b' > 'B', 'c' > 'C', 'd' > 'D'

There are two line of code have bugs. please find them and correct them.

1   
  
C:\Users\Yu\Documents\Tencent Files\651412330\Image\C2C\`MILZW444]ZCOU{A37{KGA3.png  
  
  
  




char a[10] = "aBC", b[10] = "ABcdaaaaa", c[80] = {'a', '\0' };

fun(a, b, c);

printf("The string a: "); puts(a);

printf("The string b: "); puts(b);

printf("The string c: "); puts(c);

void fun(char \*p, char \*q, char \*c)

{

int k = 0;

while (\*p !='\0' || \*q !='\0' )

{

if (\*p < \*q)

{

c[k] = \*q;

}

else

{

c[k] = \*p;

}

k++;

if (\*p)

{

p++;

}

if (\*q)

{

q++;

}

}

}