1. It makes a pointer to an int object, named ptr and makes it to point to the place in the memory where the object(should be an integer) named  var is stored

Practical example:  C++ Coding task #1 from the same homework

2. while first checks if the condition is met and then initiates the body of the loop, while the do-while first goes trough the body of the loop and then if the while condition is met it goes trough it again

Practical example:  while(age >= 18){sellAlcohool();}
do{requireNextFieldOfInfo();}while(!maxFieldsOfInfo) // да кажем че първото поле е за име, винаги ще се въвежда, без значение коя инстанция или колко ни е сегашния брой полета с информация за запълване

3. for ( int i  =  1; i <= maxIterations; i++){ cout << " Loop number:  " << i << endl;}

Предполагаме че сме задали променлива от тип integer и сме я инициализирали с цяло число. Цикъла ще се итерира maxIterations-1  пъти и ще изпълни maxIterations пъти тялото, което принтира в конзолата "Loop number: " и след него номера на изпълнение, определен от моментната стойност на i и след това минава на нов ред.

Practical Example:  когато искаме да обходим колекция от данни, както в Department::displayEmployees() от C++ Coding task #2

4. Много често се налага да има фрагменти код които да се преизползват или се използват за една и съща функционалност/процедурност. Затова функциите са полезни, да заключат тази функционалност под едно име, с лесно извикване, вместо да се имплементира цялата логика всеки път.

```
double square(double n)
{
	return n*n;
}
```

Practical Example:  Ако искаме display функция, отпечатваща полетата на даден обект, вместо секи път да ги принтираме поле по поле в мейна, за всеки обект или да трябва да правим цъкъл за това там

5. It is good when we make an object with new and call the constructor, to call a destructor after to clear the memory

Differences Between Stack and Heap Allocations

- In a stack, the allocation and de-allocation are automatically done by the compiler whereas, in heap, it needs to be done by the programmer manually.
- Handling the Heap frame is costlier than handling the stack frame.
- Memory shortage problem is more likely to happen in stack whereas the main issue in heap memory is fragmentation.
- Stack frame access is easier than the heap frame as the stack has a small region of memory and is cache-friendly but in the case of heap frames which are dispersed throughout the memory so it causes more cache misses.

- A stack is not flexible, the memory size allotted cannot be changed whereas a heap is flexible, and the allotted memory can be altered.
- Accessing the time of heap takes is more than a stack.

Resources: https://www.geeksforgeeks.org/stack-vs-heap-memory-allocation/

6. The meaning of Encapsulation, is to make sure that "sensitive" data is hidden from users. To achieve this, you must declare class variables/attributes as private (cannot be accessed from outside the class).

Source: Google

Practical Example: We have a field in the class bankAccount named balanced. Obviously we don't want in any way to let the outside world have any access to that field, so they can not manipulate it to make it have any amount of money. We would just put that field in the private part and make a method modifyBalance(double amount) in the public part

7. Exception handling in C++ is done using the try and catch keywords. To catch exceptions, a portion of code is placed under inspection. This is done by enclosing this portion of code in a try block. When an exception occurs within the try block, control is transferred to the exception handler.

Source: Google

```
try {
  int age = 15;
  if (age >= 18) {
    cout << "Access granted - you are old enough.";
  } else {
    throw (age);
  }
}
catch (int myNum) {
  cout << "Access denied - You must be at least 18 years old.\n";
  cout << "Age is: " << myNum;
}
```

Source: https://www.w3schools.com/cpp/cpp_exceptions.asp

8. Векторите са контейнери за данни, които динамично си променят големината според нуждата, докато масивите трябва да имат предварително зададена големина, което не винаги е оптимално за случая. Използваме ги когато искаме да имаме някаква колекция от данни или указатели към данни

9. Разлики между референции и препратки

**Tabular form of difference between References and Pointers in C++**

| | References | Pointers |
|---|---|---|
| **Reassignment** | The variable cannot be reassigned in Reference. | The variable can be reassigned in Pointers. |
| **Memory Address** | It shares the same address as the original variable. | Pointers have their own memory address. |
| **Work** | It is referring to another variable. | It is storing the address of the variable. |
| **Null Value** | It does not have null value. | It can have value assigned as null. |
| **Arguments** | This variable is referenced by the method pass by value. | The pointer does it work by the method known as pass by reference. |

Scenarios where one might be preffered over the other:

The performances are exactly the same as references are implemented internally as pointers. But still, you can keep some points in your mind to decide when to use what:

Use references:
In function parameters and return types.
Use pointers:
If pointer arithmetic or passing a NULL pointer is needed. For example, for arrays (Note that accessing an array is implemented using pointer arithmetic).
To implement data structures like a linked list, a tree, etc. and their algorithms. This is so because, in order to point to different cells, we have to use the concept of pointers.

Source: https://www.geeksforgeeks.org/pointers-vs-references-cpp/

10.

Templates are the basis for generic programming in C++. As a strongly-typed language, C++ requires all variables to have a specific type, either explicitly declared by the programmer or deduced by the compiler. However, many data structures and algorithms look the same no matter what type they are operating on.

Templates in C++ act as the foundation of generic programming. It is a simple yet powerful tool that acts as a blueprint for creating generic functions or classes. While using a template in C++, you pass a data type as a parameter.

```
1. template<class T> T add(T &a,T &b)
2. {
3.     T result = a+b;
4.     return result;
5.
6. }
```

Source: Google and https://www.javatpoint.com/cpp-templates