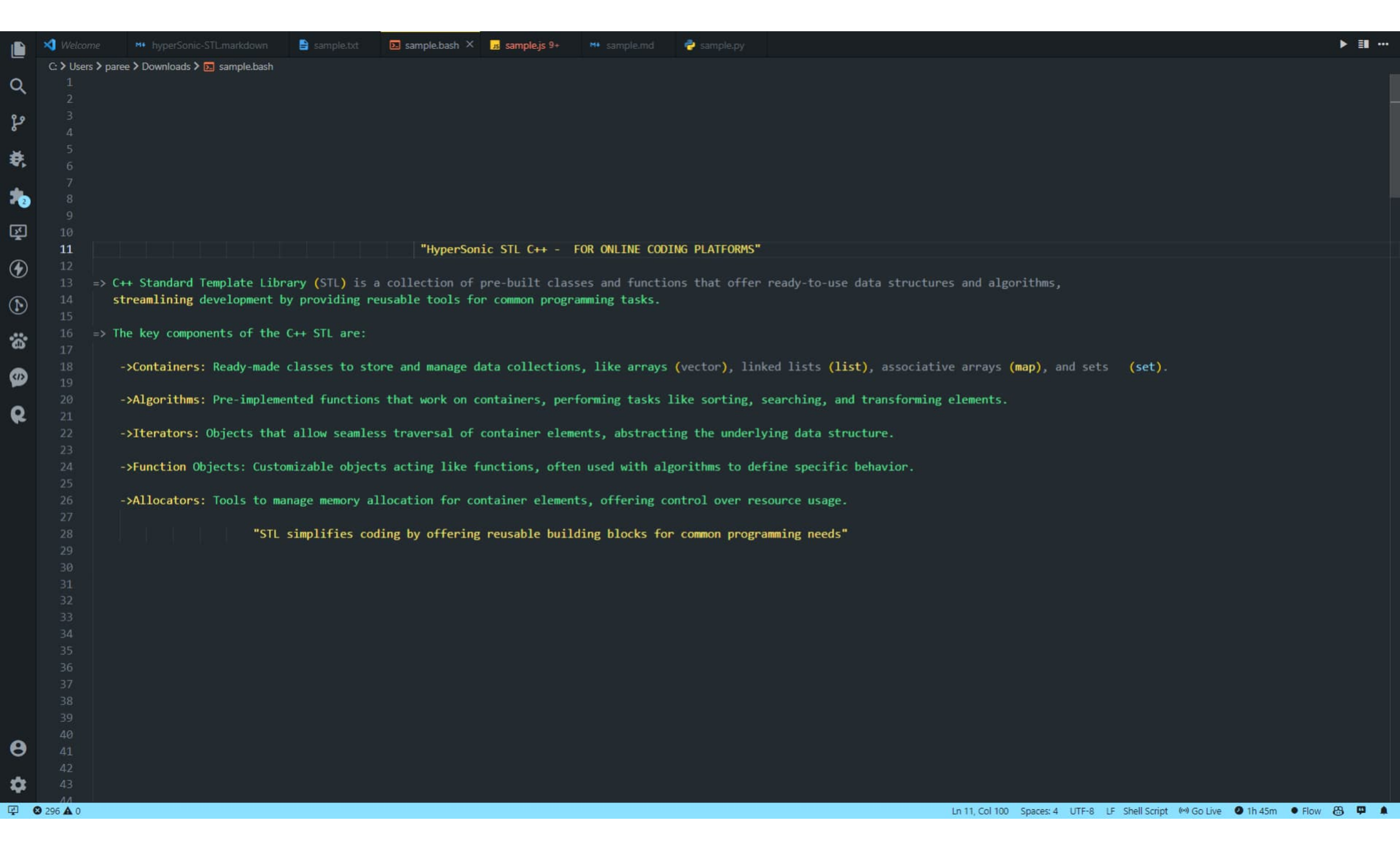
Welcome
hyperSonic-STL.markdown
sample.txt
sample.bash
sample.js 9+
sample.md
sample.py
C: > Users > paree > Downloads > sample.bash
"HyperSonic STL C++ - FOR ONLINE CODING PLATFORMS"
=> C++ Standard Template Library (STL) is a collection of pre-built classes and functions that offer ready-to-use data structures and algorithms,
streamlining development by providing reusable tools for common programming tasks.
=> The key components of the C++ STL are:
->Containers: Ready-made classes to store and manage data collections, like arrays (vector), linked lists (list), associative arrays (map), and sets (set).
->Algorithms: Pre-implemented functions that work on containers, performing tasks like sorting, searching, and transforming elements.
->Iterators: Objects that allow seamless traversal of container elements, abstracting the underlying data structure.
->Function Objects: Customizable objects acting like functions, often used with algorithms to define specific behavior.
->Allocators: Tools to manage memory allocation for container elements, offering control over resource usage.
"STL simplifies coding by offering reusable building blocks for common programming needs"
296 0
Ln 11, Col 100
Spaces: 4
UTF-8
LF
Shell Script
Go Live
1h 45m
Flow

```
48
49
50
51
52  //C++ vectors : Dynamic size continoius storage containers allows constant time for inserting at end using resizing strategy.
53
54  Header file -> #include <vector>
55
56  initializing and declaring a vector :                    vector<int> v(size,value_to_initialize);
57
58  with an array if int array[3]={7,8,9}; then              vector<int> v(arr,arr+3);
59
60  with another vector then                                 vector<int> v(v1.begin(),v1.end());
61
62  resizing a vector                                        v.resize(10) or v.resize(10,-1); //with initializing all values to -1
63
64  filling a vector with consecutive values                 iota(v.begin(),vector.begin()+3,-1);  //fills it with consecutive elements as :-1, 0, 1
65
66  inserting into vector                                    v.insert(v.begin(),value_to_insert);
67
68  erasing from vector                                      v.erase(v.begin()+3) or v.erase(v.begin()+3,v.end());
69
70  clear or fill into vector                                v.clear() and   fill(v.begin(),v.begin()+3,-1);
71
72  finding max and min element from vector                  int min= *min_element(v.begin(),v.end()); or int max= *max_element(v.begin(),v.end());
73
74  swapping two vectors                                     v.swap(other_vector);
75
76  utility functions                                        bool status= v.empty(); v.push_back(1); v.pop_back(); int n= v.size();
77
78  finding an element                                       vector<int>:: iterator it= find(v.begin(),v.end(),value);
79
80  Summing elements withing a range                         int sum= accumulate(v.begin(),v.begin()+3, intial_value_of_sum);
81
82  count occurence of element                               int count= count(v.being(),v.end(),value);
83
84
85
86
87
88
89
90
```

```
//C++ Mathematics

Header file -> #include <cmath>

    double x = 2.0;
    double y = 3.0;

    // Trigonometric functions
    double sin_result = std::sin(x);       // Sine
    double cos_result = std::cos(x);       // Cosine
    double tan_result = std::tan(x);       // Tangent

    // Exponential function
    double exp_result = std::exp(x);       // Exponential function (e^x)

    // Square root function
    double sqrt_result = std::sqrt(x);     // Square root

    double floor_result = std::floor(x);   // Round down to nearest integer
    double fmod_result = std::fmod(x, y);  // Floating-point remainder of x/y

    // Logarithmic functions
    double log_result = std::log(x);       // Natural logarithm
    double log10_result = std::log10(x);   // Base-10 logarithm

    // Power functions
    double pow_result = std::pow(x, y);    // x raised to the power of y

    // Other functions
    double round_result = std::round(x);   // Round to nearest integer
    double abs_result = std::abs(x);       // Absolute/positive value
    double trunc_result = std::trunc(x);   // Round towards zero
```

```
//C++ Algorithms

Header file -> #include <algorithm>

    std::vector<int> numbers = {5, 2, 8, 3, 1, 7};

    // 1. std::sort                                                    // 2. std::reverse
    std::sort(numbers.begin(), numbers.end());                         std::reverse(numbers.begin(), numbers.end());

    // 3. std::max_element                                             // 4. std::min_element
    auto maxIt = std::max_element(numbers.begin(), numbers.end());     auto minIt = std::min_element(numbers.begin(), numbers.end());

    // 5. std::accumulate                                              // 6. std::count
    int sum = std::accumulate(numbers.begin(), numbers.end(), 0);      int countOf2 = std::count(numbers.begin(), numbers.end(), 2);

    // 7. std::find                                                    // 8. std::replace
    auto findIt = std::find(numbers.begin(), numbers.end(), 8);        std::replace(numbers.begin(), numbers.end(), 2, 10);

    // 11. std::for_each
    #here we are using lambda function as [](int num) which has no name and we are passing it as a parameter to for_each function
    std::for_each(numbers.begin(), numbers.end(), [](int num){std::cout << num << " ";});

    //12. std::transform
    std::transform(numbers.begin(), numbers.end(), numbers.begin(), [](int num) {
        return num * num;
    });

    // 13. std::binary_search                                          // 15. std::shuffle
    bool found = std::binary_search(numbers.begin(), numbers.end(), 5);  std::random_shuffle(numbers.begin(), numbers.end());

    // 16. std::merge
    std::vector<int> mergedVector;
    std::merge(numbers.begin(), numbers.end(), copyVector.begin(), copyVector.end(), std::back_inserter(mergedVector));

    // 17. std::nth_element for any type of container
    std::nth_element(numbers.begin(), numbers.begin() + 3, numbers.end());

    // 20. std::next_permutation                                       // 21. std::prev_permutation
    std::next_permutation(numbers.begin(), numbers.end());             std::prev_permutation(numbers.begin(), numbers.end());

    // 22. std::lower_bound                                            // 23. std::upper_bound
    auto lowerIt = std::lower_bound(numbers.begin(), numbers.end(), 5);  auto upperIt = std::upper_bound(numbers.begin(), numbers.end(), 5);

    // 24. std::rotate                                                 // 25. std::is_sorted
    std::rotate(numbers.begin(), numbers.begin() + 3, numbers.end());  bool isSorted = std::is_sorted(numbers.begin(), numbers.end());
```

```
//C++ Strings

Header file -> #include <string>

    std::string str = "Hello, world!";

    // std::substr
    std::string substr = str.substr(7, 5); // Extract "world"

    // std::replace
    std::replace(str.begin(), str.end(), 'o', 'x'); // Replace 'o' with 'x'

    // std::insert
    str.insert(5, " there"); // Insert " there" at position 5

    // std::substr and std::find together for extraction
    std::string word = str.substr(pos, 5); // Extract "world"

    // std::stoi, std::stof, std::stod, std::stol, std::stoll
    int intValue = std::stoi("42"); // Convert string to int

    // std::toupper, std::tolower
    std::string upperStr = str;
    std::transform(upperStr.begin(), upperStr.end(), upperStr.begin(), ::toupper);

//Dynamic memory allocation

char *str=(char *)malloc(sizeof(char)*10);
char *str=(char *)calloc(10,sizeof(char));

*str=(char *)realloc(str,20*sizeof(char));

//c++ string to c string

string my_string="hello";
char *c_string= my_str.c_str();

// c string to c++ string

char *c_string="hello";
string my_string=string(c_string);

    // std::find
    std::size_t pos = str.find("world"); // Find position of "world"

    // std::erase
    str.erase(5, 2); // Erase ", "

    // std::compare
    int compareResult = str.compare("Hello, x there!");

    // std::c_str
    const char* cStr = str.c_str(); // Get C-style string

    // std::to_string
    std::string intStr = std::to_string(42); // Convert int to string

    // std::find_first_of, std::find_first_not_of,
    std::size_t foundPos = str.find_first_of("aeiou"); // Find first vowel
```