МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №9**

по дисциплине: Объектно-ориентированное программирование Тема: Использование стандартной библиотеки шаблонов STL.

Выполнил: студент группы ПВ-223

Дмитриев А.А.

Проверил:

Черников С.В.

Белгород 2024 г.

**Цель работы:** Знакомство со стандартной библиотекой шаблонов в С++; получение навыков использования классов контейнеров, итераторов, алгоритмов.

**Задание:**

Разработать программное обеспечения для решения

соответствующего варианта. Оформить отчет. Для реализации

поставленных задач требуется использовать следующие библиотеки

классов: list, vector, queue, iostream, algorithm, set, iterator, map, stack.

2 вариант: Разработать программное обеспечение для решения следующей задачи: организовать поиск по html странице. Построить словарь всех слов встречающихся на заданной html страницы в следующем виде map &lt;key, value&gt;, где value представляет собой объект класса с поля количества встречаемости слов и ссылки на них, key искомое слово. Организовать неточный поиск, использовать ::iterator.

#include <map>

#include <string>

#include <fstream>

#include <iostream>

#include <iterator>

#include <sstream>

#include <vector>

class AmountIndex

{

private:

int amount;

std::vector<int> indexes;

public:

AmountIndex() {}

AmountIndex(int amount, std::vector<int> indexes) {

this->amount = amount;

this->indexes = indexes;

}

int getAmount() {

return amount;

}

std::vector<int> getIndexes() {

return indexes;

}

void setAmount(int amount) {

this->amount = amount;

}

void setIndexes(std::vector<int> indexes) {

this->indexes = indexes;

}

void addAmount() {

(this->amount)++;

}

void addIndex(int i) {

indexes.push\_back(i);

}

friend std::ostream& operator<<(std::ostream& os, const AmountIndex& ai)

{

os << "am: " << ai.amount << " ind: ";

for (auto index : ai.indexes) {

std::cout << index << ' ';

}

std::cout << '\b';

return os;

}

~AmountIndex() {}

};

class HtmlFinder

{

private:

std::map<std::string, AmountIndex> dict;

public:

HtmlFinder() {}

HtmlFinder(std::string html) {

bool ignore = false;

bool isBackSlash = false;

std::string buf;

for (int i = 0; i < html.length(); i++)

{

char c = html[i];

if (c == '<') {

ignore = true;

}

else if (c == '>') {

ignore = false;

continue;

}

if (ignore) {

continue;

}

else if (!isalpha(c)) {

if (buf.empty())

continue;

for (int j = 0; j < buf.length(); j++) {

buf[j] = tolower(buf[j]);

}

std::string temp(buf);

if (dict.find(temp) != dict.end()) { // todo maybe truble

AmountIndex ai = dict[temp];

ai.addAmount();

ai.addIndex(i);

}

else {

std::vector<int> indeces;

indeces.push\_back(i);

dict[temp] = AmountIndex(1, indeces);

}

buf.clear();

continue;

}

buf += c;

}

}

std::map<std::string, AmountIndex> getDictionary() {

return dict;

}

AmountIndex find(std::string word) {

std::map<std::string, AmountIndex>::iterator iter = dict.begin();

while (iter != dict.end()) {

if ((\*iter).first == word) {

return (\*iter).second;

}

iter++;

}

AmountIndex res;

return res;

}

~HtmlFinder() {}

};

int main() {

std::ifstream f("text.html");

std::stringstream ss;

ss << f.rdbuf();

std::string text = ss.str();

HtmlFinder htmlf(text);

for (auto pair : htmlf.getDictionary())

{

std::cout << pair.first << ' ' << pair.second << '\n';

}

std::cout << "\nresult of foundation:\n";

std::cout << "while - " << htmlf.find("while") << '\n';

std::cout << "a - "<< htmlf.find("a") << '\n';

return 0;

}

Тестовые данные

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Hash Maps in C++ with Simple Code Examples and Explanations</title>

<meta name="description" content="Code examples of how hash maps work in C++ with simple explanations. The C++ standard library&#39;s &#34;std::unordered\_map&#34;.">

<link rel="shortcut icon" href="/favicon.png">

<link rel="apple-touch-icon" href="/icon-192.png">

<link rel="icon" type="image/png" sizes="192x192" href="/icon-192.png">

<meta property="og:site\_name" content="C++ By Example" />

<meta property="og:title" content="Hash Maps" />

<meta property="og:description" content="Code examples of how hash maps work in C++ with simple explanations. The C++ standard library&#39;s &#34;std::unordered\_map&#34;." />

<meta property="og:type" content="website" />

<meta property="og:url" content="https://cppbyexample.com/hash\_map.html" />

<meta property="og:image" content="https://cppbyexample.com/icon-512.png" />

<style>

:root {

/\* Solarized \*/

--base03: #002b36;

--base02: #073642;

--base01: #586e75;

--base00: #657b83;

--base0: #839496;

--base1: #93a1a1;

--base2: #eee8d5;

--base3: #fdf6e3;

--yellow: #b58900;

--orange: #cb4b16;

--red: #dc322f;

--magenta: #d33682;

--violet: #6c71c4;

--blue: #268bd2;

--cyan: #2aa198;

--green: #859900;

}

body {

-webkit-text-size-adjust: none;

margin: 0 auto;

padding: 0;

font-family: Verdana, "Bitstream Vera Sans", sans-serif;

line-height: 1.5em;

text-align: left;

word-wrap: break-word;

}

h1 {

font-family: Helvetica, Georgia, sans-serif;

font-size: 1.7em;

line-height: 1.5em;

text-align: left;

margin-bottom: 0em;

}

#tags {

margin-top: 0em;

}

h2 {

font-family: Helvetica, Georgia, sans-serif;

font-size: 1.5em;

line-height: 1.5em;

text-align: left;

}

img {

max-width: 100%;

}

#title {

font-family: Helvetica, Georgia, sans-serif;

font-size: 2em;

font-weight: bold;

text-decoration: none;

border-width: 0 0 0 0;

border-style: none none none none;

display: inline;

}

#banner {

padding: 40px 00px 0px 0px;

}

#banner p {

font-size: 0.9em;

}

nav {

margin: 1em;

text-decoration: none;

}

section#articles, #banner {

position: relative;

display: block;

width: 620px;

margin: 0 auto;

}

/\* 500-pixel layout for smaller screens \*/

@media (max-width: 680px) {

section#articles, #banner {

width: calc(100vw - 30px);

}

}

a:link, a:visited {

border-width: 0 0 1px 0;

padding: 3px 0px 2px 0px;

background-color: inherit;

color: inherit;

}

a.tag, code.tag {

color: var(--cyan);

padding: 0.1em 0.3em 0.1em 0.3em;

border-radius: 10px;

}

pre {

font-family: Monaco, monospace;

font-size: 0.95em;

white-space: pre-wrap;

padding: 0.5em;

line-height: 1.5em;

border-radius: 10px;

}

code {

color: var(--blue);

padding: 0.1em 0.3em 0.1em 0.3em;

border-radius: 10px;

font-size: 1.2em;

}

@media (prefers-color-scheme: light) {

body {

color: #222222;

background-color: var(--base3);

}

::selection {

background-color: var(--base2);

}

/\* Sublime Dark \*/

pre {

color: var(--base0);

background-color: var(--base03);

}

pre::selection, pre > span::selection {

background-color: var(--base02);

}

/\* Sublime Dark \*/

code {

background-color: var(--base2);

}

.Comment, .c, .ch, .cm, .c1, .cs {

color: var(--base01);

}

}

@media (prefers-color-scheme: dark) {

body {

color: #dddddd;

background-color: var(--base03);

}

::selection {

background-color: var(--base02);

}

/\* Sublime Light \*/

pre {

color: var(--base00);

background-color: var(--base3);

}

pre::selection, pre > span::selection {

background-color: var(--base2);

}

/\* Sublime Light \*/

code {

background: var(--base02);

}

.Comment, .c, .ch, .cm, .c1, .cs {

color: var(--base1);

}

}

.PreProc, .cp {

color: var(--orange);

}

.LineNr {

color: var(--base1);

}

.Constant, .cpf, .mi, .s {

color: var(--cyan);

}

.Type, .kt {

color: var(--yellow);

}

.Special, .se {

color: var(--red);

}

.Statement, .k {

color: var(--green);

}

.dateline {

color: #aaa;

font-size: .75em;

margin-top: -2em;

margin-bottom: 1.5em;

font-weight: normal;

}

.footer {

margin-top: 14em;

padding-bottom: 1em;

text-align: left;

background-color: transparent;

}

.smallprint {

margin-top: 4em;

line-height: 1.8em;

font-size: .8em;

text-align: left;

color: #aaa;

}

</style>

<!-- Global site tag (gtag.js) - Google Analytics -->

<script async src="https://www.googletagmanager.com/gtag/js?id=G-MB9C7KPQHY"></script>

<script>

window.dataLayer = window.dataLayer || [];

function gtag(){dataLayer.push(arguments);}

gtag('js', new Date());

gtag('config', 'G-MB9C7KPQHY');

</script>

<!-- Google AdSense -->

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js?client=ca-pub-7449174967368098"

crossorigin="anonymous"></script>

</head>

<body>

<div id="banner">

<a id="title" href="/">C++ By Example</a>

<nav>

<a href="/">Home</a>

|

<a href="/all\_tags.html">All Tags</a>

|

<a href="/cdn-cgi/l/email-protection#74171b1a0015170034170404160d110c15190418115a171b19">Contact</a>

|

<a href="/about.html">About</a>

</nav>

</div>

<hr />

<section id="articles">

<article>

<h1>Hash Maps</h1>

<h4 id="tags">

<code><a class="tag" href="/tags/c++17.html">c++17</a></code>

<code><a class="tag" href="/tags/containers.html">containers</a></code>

<code><a class="tag" href="/tags/intermediate.html">intermediate</a></code>

</h4>

<p><strong>Related:</strong> <a href="/hash\_set.html">Hash Sets</a></p>

<p>

Hash maps, sometimes called dictionary or table, are known as <em>unordered maps</em> in C++.

The <a href="/what\_is\_std.html">C++ standard library&rsquo;s</a> implementation of hash map is called <code>std::unordered\_map</code>.

<code>std::unordered\_map</code> makes no guarantees about the order of its keys and their order can depend on when they are inserted into the map.

This means when iterating the key-value pairs of <code>std::unordered\_map</code> we cannot know the order of iteration.

This allows <code>std::unordered\_map</code> to optimize for the usage of element insertion and search, or the finding of individual values by key, when compared to <a href="/map.html">std::map</a>.

</p>

<pre><span class="cp">#include</span><span class="w"> </span><span class="cpf">&lt;unordered\_map&gt;</span><span class="cp"></span>

<span class="cp">#include</span><span class="w"> </span><span class="cpf">&lt;string&gt;</span><span class="cp"></span>

<span class="cp">#include</span><span class="w"> </span><span class="cpf">&lt;iostream&gt;</span><span class="cp"></span>

<span class="kt">int</span><span class="w"> </span><span class="nf">main</span><span class="p">()</span><span class="w"> </span><span class="p">{</span><span class="w"></span>

<span class="w"> </span><span class="n">std</span><span class="o">::</span><span class="n">unordered\_map</span><span class="o">&lt;</span><span class="kt">int</span><span class="p">,</span><span class="w"> </span><span class="n">std</span><span class="o">::</span><span class="n">string</span><span class="o">&gt;</span><span class="w"> </span><span class="n">statusMessages</span><span class="p">{</span><span class="w"></span>

<span class="w"> </span><span class="p">{</span><span class="mi">200</span><span class="p">,</span><span class="w"> </span><span class="s">&quot;Success&quot;</span><span class="p">},</span><span class="w"></span>

<span class="w"> </span><span class="p">{</span><span class="mi">404</span><span class="p">,</span><span class="w"> </span><span class="s">&quot;This is not the page you&#39;re looking for&quot;</span><span class="p">},</span><span class="w"></span>

<span class="w"> </span><span class="p">{</span><span class="mi">403</span><span class="p">,</span><span class="w"> </span><span class="s">&quot;Unauthorized&quot;</span><span class="p">},</span><span class="w"></span>

<span class="w"> </span><span class="p">{</span><span class="mi">418</span><span class="p">,</span><span class="w"> </span><span class="s">&quot;I&#39;m a teapot&quot;</span><span class="p">},</span><span class="w"></span>

<span class="w"> </span><span class="p">};</span><span class="w"></span>

<span class="w"> </span><span class="n">statusMessages</span><span class="p">.</span><span class="n">insert</span><span class="p">({</span><span class="mi">503</span><span class="p">,</span><span class="w"> </span><span class="s">&quot;Something went wrong&quot;</span><span class="p">});</span><span class="w"></span>

<span class="w"> </span><span class="n">std</span><span class="o">::</span><span class="n">cout</span><span class="w"> </span><span class="o">&lt;&lt;</span><span class="w"> </span><span class="n">statusMessages</span><span class="p">[</span><span class="mi">418</span><span class="p">]</span><span class="w"> </span><span class="o">&lt;&lt;</span><span class="w"> </span><span class="s">&quot;</span><span class="se">\n</span><span class="s">&quot;</span><span class="p">;</span><span class="w"> </span>

<span class="p">}</span><span class="w"></span>

</pre>

<pre>I'm a teapot</pre>

<script data-cfasync="false" src="/cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js"></script>

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js?client=ca-pub-7449174967368098"

crossorigin="anonymous"></script>

<p>

<ins class="adsbygoogle"

style="display:block; text-align:center;"

data-ad-layout="in-article"

data-ad-format="fluid"

data-full-width-responsive="true"

data-ad-client="ca-pub-7449174967368098"

data-ad-slot="3409050717"></ins>

</p>

<script>

(adsbygoogle = window.adsbygoogle || []).push({});

</script>

<h2 id="element-lookup">Element Lookup</h2>

<p>

There&rsquo;s two methods to find an element in a <code>std::unordered\_map</code>: the <code>find()</code> method and the square bracket operator(<code>[]</code>).

To learn more <a href="/hash\_map\_find.html">click here</a>.

</p>

<h2 id="storing-custom-types">Storing Custom Types</h2>

<p>

<code>std::unordered\_map</code> requires that the keys of the map are able to be compared for equality, meaning the keys should support the equality operator (<code>==</code>).

The keys of <code>std::unordered\_map</code> are also required to be able to be hashable meaning you need to write a specialization of <code>std::hash</code> for the key type.

To learn more about storing custom types in <code>std::unordered\_map</code> <a href="/hash\_map\_custom\_types.html">click here</a>.

</p>

<h2 id="another-key-value-container">Another Key-Value Container</h2>

<p>

There&rsquo;s another <em>ordered</em> key-value container in C++: <code>std::map</code>.

While technically not a hash map, <code>std::map</code> is a key-value container that maintains its keys in sorted order at all times.

Click <a href="/map.html">here</a> to learn more.

</p>

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js?client=ca-pub-7449174967368098"

crossorigin="anonymous"></script>

<p>

<ins class="adsbygoogle"

style="display:block; text-align:center;"

data-ad-layout="in-article"

data-ad-format="fluid"

data-full-width-responsive="true"

data-ad-client="ca-pub-7449174967368098"

data-ad-slot="3409050717"></ins>

</p>

<script>

(adsbygoogle = window.adsbygoogle || []).push({});

</script>

</article><hr />

<p>For more C++ By Example, <a href="/index.html">click here</a>.</p>

<div id="footer">

<form id="siteSearch" action="https://duckduckgo.com/" method="get" style="margin-bottom: 2.5em;" onsubmit="this.q.value = 'site:cppbyexample.com ' + this.q.value">

<div>

<input name="q" type="text" 10 style="margin-right: 8px; width: 66%;" />

<input type="submit" value="Search" />

</div>

</form>

<p class="smallprint">

Copyright &copy; 2022 Rob Cusimano.

All code is owned by Rob Cusimano and made available under the following <a href="/license.html">MIT license</a>.

</p>

</div>

</section>

</body>

</html>

Пример работы

a am: 11 ind: 1315 1317 3290 3298 3471 10488 11523 12476 13092 13558 13595

aa am: 1 ind: 1586

aaa am: 2 ind: 5952 6422

able am: 2 ind: 12858 13049

about am: 3 ind: 7349 8201 13187

adjust am: 1 ind: 1683

adsbygoogle am: 4 ind: 12278 12299 14372 14393

align am: 5 ind: 1866 2074 2335 6195 6391

all am: 3 ind: 7155 13662 15062

allows am: 1 ind: 8454

also am: 1 ind: 13029

an am: 1 ind: 12463

and am: 5 ind: 200 8227 8537 12544 15096

another am: 2 ind: 13371 13450

are am: 4 ind: 7935 8267 12853 13024

arguments am: 1 ind: 6696

articles am: 2 ind: 2949 3201

as am: 1 ind: 7944

at am: 1 ind: 13658

auto am: 2 ind: 1717 3073

available am: 1 ind: 15111

b am: 4 ind: 1163 1256 1404 1438

background am: 12 ind: 3406 4129 4218 4366 4490 4607 4877 4967 5118 5241 5358 6225

banner am: 4 ind: 2713 2786 2958 3210

base am: 23 ind: 1154 1185 1216 1247 1278 1308 1338 1368 4147 4236 4336 4384 4508 4625 4724 4895 4985 5087 5136 5259 5370 5470 5615

bd am: 1 ind: 1559

be am: 3 ind: 12864 13044 13055

bitstream am: 1 ind: 1787

block am: 1 ind: 3019

blue am: 2 ind: 1551 3882

body am: 3 ind: 1644 4068 4816

bold am: 1 ind: 2539

border am: 6 ind: 2594 2629 3327 3584 3812 3949

bottom am: 3 ind: 2107 6039 6166

bracket am: 1 ind: 12563

break am: 1 ind: 1902

by am: 4 ind: 7041 8584 14484 15079

c am: 14 ind: 168 1527 1530 4668 4682 5414 5428 6751 7036 7553 7973 8026 13492 14479

calc am: 1 ind: 3240

called am: 2 ind: 7910 8094

can am: 1 ind: 8243

cannot am: 1 ind: 8397

cb am: 1 ind: 1436

ch am: 2 ind: 4673 5419

click am: 4 ind: 12656 13287 13691 14521

cm am: 2 ind: 4678 5424

code am: 6 ind: 187 3481 3850 4578 5329 15067

color am: 30 ind: 3412 3440 3505 3870 4034 4092 4135 4224 4324 4372 4496 4613 4712 4783 4840 4883 4973 5075 5124 5247 5458 5540 5603 5682 5746 5815 5882 5946 6231 6416

comment am: 2 ind: 4664 5410

compared am: 2 ind: 8603 12873

config am: 1 ind: 6741

constant am: 1 ind: 5647

contact am: 1 ind: 7291

container am: 3 ind: 13396 13487 13615

containers am: 1 ind: 7656

copy am: 1 ind: 15022

copyright am: 1 ind: 15016

cout am: 1 ind: 11071

cp am: 1 ind: 5520

cpf am: 1 ind: 5653

cs am: 2 ind: 4688 5434

cusimano am: 2 ind: 15041 15092

custom am: 2 ind: 12740 13202

cyan am: 3 ind: 1580 3517 5694

d am: 2 ind: 1347 1495

dark am: 3 ind: 4281 4558 4796

datalayer am: 3 ind: 6627 6646 6681

date am: 1 ind: 6721

dateline am: 1 ind: 5926

dc am: 1 ind: 1464

dddddd am: 1 ind: 4849

decoration am: 2 ind: 2568 2906

depend am: 1 ind: 8250

dictionary am: 1 ind: 7921

display am: 2 ind: 2677 3012

e am: 2 ind: 1225 1377

eee am: 1 ind: 1345

element am: 3 ind: 8523 12382 12471

em am: 30 ind: 1842 2018 2050 2112 2168 2279 2311 2508 2823 2877 3546 3552 3558 3564 3697 3760 3792 3911 3917 3923 3929 3992 5982 6012 6046 6138 6171 6306 6338 6367

equality am: 2 ind: 12886 12932

example am: 2 ind: 7053 14492

examples am: 1 ind: 196

explanations am: 1 ind: 221

f am: 1 ind: 1468

family am: 5 ind: 1766 1956 2217 2448 3647

fdf am: 1 ind: 1375

find am: 2 ind: 12460 12524

finding am: 1 ind: 8560

following am: 1 ind: 15131

font am: 15 ind: 1759 1949 2006 2210 2267 2441 2498 2526 2811 3640 3684 3980 5970 6064 6356

footer am: 1 ind: 6107

for am: 6 ind: 3117 8502 10047 12877 13137 14472

function am: 1 ind: 6664

g am: 1 ind: 6746

georgia am: 3 ind: 1976 2237 2468

green am: 2 ind: 1610 5895

gt am: 4 ind: 8757 8877 8999 9522

gtag am: 3 ind: 6669 6706 6733

guarantees am: 1 ind: 8195

h am: 2 ind: 1929 2190

hash am: 7 ind: 158 7453 7837 7887 8080 13133 13563

hashable am: 1 ind: 13064

height am: 5 ind: 1835 2043 2304 3785 6331

helvetica am: 3 ind: 1967 2228 2459

here am: 4 ind: 12665 13296 13720 14530

home am: 1 ind: 7100

i am: 2 ind: 10480 11519

img am: 1 ind: 2365

implementation am: 1 ind: 8072

in am: 6 ind: 166 7971 12474 13211 13490 13642

include am: 3 ind: 8711 8838 8958

individual am: 1 ind: 8574

inherit am: 2 ind: 3421 3449

inline am: 1 ind: 2685

insert am: 1 ind: 10762

inserted am: 1 ind: 8276

insertion am: 1 ind: 8533

int am: 2 ind: 9076 9391

intermediate am: 1 ind: 7748

into am: 1 ind: 8281

iostream am: 1 ind: 8996

is am: 4 ind: 8087 10011 13593 15070

iterating am: 1 ind: 8332

iteration am: 1 ind: 8425

its am: 2 ind: 8218 13634

js am: 1 ind: 6710

k am: 1 ind: 5862

key am: 6 ind: 8340 8588 13145 13375 13471 13599

keys am: 5 ind: 8223 12838 12904 12985 13639

know am: 1 ind: 8402

known am: 1 ind: 7941

kpqhy am: 1 ind: 6757

kt am: 1 ind: 5726

layout am: 1 ind: 3113

learn am: 3 ind: 12615 13176 13729

left am: 5 ind: 1872 2080 2341 6201 6397

library am: 1 ind: 8045

license am: 1 ind: 15171

light am: 3 ind: 4048 5032 5309

line am: 5 ind: 1828 2036 2297 3778 6324

linenr am: 1 ind: 5583

link am: 1 ind: 3295

looking am: 1 ind: 10043

lookup am: 1 ind: 12394

lt am: 8 ind: 8740 8867 8987 9340 11098 11102 11289 11293

m am: 2 ind: 10486 11521

made am: 1 ind: 15101

magenta am: 1 ind: 1491

main am: 1 ind: 9128

maintains am: 1 ind: 13630

makes am: 1 ind: 8181

map am: 17 ind: 8084 8126 8175 8289 8387 8486 8639 8754 9337 12508 12815 12849 13020 13243 13517 13567 13590

maps am: 4 ind: 163 7463 7892 7968

margin am: 9 ind: 1709 2100 2159 2872 3065 6002 6032 6128 6297

max am: 2 ind: 2383 3156

mb am: 1 ind: 6749

meaning am: 2 ind: 12895 13072

means am: 1 ind: 8317

media am: 3 ind: 3151 4019 4768

method am: 1 ind: 12540

methods am: 1 ind: 12452

mi am: 1 ind: 5658

mit am: 1 ind: 15159

monaco am: 1 ind: 3655

monospace am: 1 ind: 3666

more am: 4 ind: 12620 13181 13734 14477

n am: 1 ind: 11396

nav am: 1 ind: 2851

need am: 1 ind: 13081

new am: 1 ind: 6716

no am: 1 ind: 8184

none am: 7 ind: 1689 2574 2641 2646 2651 2656 2912

normal am: 1 ind: 6079

not am: 2 ind: 10015 13556

of am: 9 ind: 8075 8214 8355 8415 8515 8563 12841 12988 13110

on am: 1 ind: 8253

operator am: 2 ind: 12572 12941

optimize am: 1 ind: 8498

or am: 2 ind: 7924 8548

orange am: 2 ind: 1431 5554

order am: 4 ind: 8211 8239 8412 13655

ordered am: 1 ind: 13467

owned am: 1 ind: 15076

p am: 1 ind: 2788

padding am: 7 ind: 1738 2735 3365 3539 3753 3904 6159

page am: 1 ind: 10024

pairs am: 1 ind: 8352

pixel am: 1 ind: 3106

position am: 1 ind: 2981

pre am: 8 ind: 3621 3727 4300 4423 4439 5051 5174 5190

prefers am: 2 ind: 4028 4777

preproc am: 1 ind: 5515

push am: 3 ind: 6686 12311 14405

px am: 15 ind: 2741 2746 2750 2754 3045 3169 3253 3342 3370 3374 3378 3382 3597 3825 3962

quot am: 12 ind: 9786 9799 10003 10052 10256 10274 10478 10500 10867 10893 11346 11401

radius am: 3 ind: 3591 3819 3956

re am: 1 ind: 10035

red am: 2 ind: 1459 5826

related am: 1 ind: 7797

relative am: 1 ind: 2991

required am: 1 ind: 13038

requires am: 1 ind: 12824

rob am: 2 ind: 15032 15083

root am: 1 ind: 1105

rsquo am: 3 ind: 8051 12438 13440

s am: 4 ind: 5662 8057 12440 13442

sans am: 5 ind: 1797 1804 1982 2243 2474

scheme am: 2 ind: 4041 4790

screens am: 1 ind: 3133

se am: 1 ind: 5795

search am: 1 ind: 8544

section am: 2 ind: 2940 3192

selection am: 6 ind: 4189 4434 4457 4938 5185 5208

serif am: 4 ind: 1810 1988 2249 2480

sets am: 1 ind: 7850

should am: 1 ind: 12911

simple am: 1 ind: 182

size am: 9 ind: 1676 2011 2272 2503 2816 3689 3985 5975 6361

smaller am: 1 ind: 3125

smallprint am: 1 ind: 6276

solarized am: 1 ind: 1132

something am: 1 ind: 10877

sometimes am: 1 ind: 7903

sorted am: 1 ind: 13649

space am: 1 ind: 3722

span am: 2 ind: 4446 5197

special am: 1 ind: 5790

specialization am: 1 ind: 13107

square am: 1 ind: 12555

standard am: 1 ind: 8037

statement am: 1 ind: 5858

statusmessages am: 3 ind: 9607 10709 11187

std am: 15 ind: 8104 8153 8365 8464 8630 9276 9465 11019 12486 12793 12998 13120 13221 13505 13578

storing am: 2 ind: 12733 13195

string am: 2 ind: 8874 9519

style am: 1 ind: 2635

sublime am: 4 ind: 4276 4553 5026 5303

success am: 1 ind: 9794

support am: 1 ind: 12919

table am: 1 ind: 7930

tag am: 2 ind: 3475 3485

tags am: 2 ind: 2138 7164

teapot am: 2 ind: 10495 11536

technically am: 1 ind: 13552

text am: 8 ind: 1671 1860 2068 2329 2557 2895 6189 6385

that am: 2 ind: 12829 13620

the am: 17 ind: 7996 8205 8285 8336 8406 8506 8552 10019 12513 12548 12833 12845 12899 12923 12980 13141 15121

their am: 1 ind: 8233

there am: 2 ind: 12432 13434

they am: 1 ind: 8263

this am: 3 ind: 8311 8447 10008

times am: 1 ind: 13668

title am: 1 ind: 2422

to am: 10 ind: 8489 8606 12455 12609 12861 13041 13052 13084 13170 13723

top am: 4 ind: 2163 6006 6132 6301

transparent am: 1 ind: 6244

two am: 1 ind: 12444

type am: 2 ind: 5721 13150

types am: 2 ind: 12751 13208

unauthorized am: 1 ind: 10269

under am: 1 ind: 15117

unordered am: 11 ind: 7958 8115 8164 8376 8475 8750 9310 12497 12804 13009 13232

usage am: 1 ind: 8512

value am: 4 ind: 8346 13381 13477 13605

values am: 1 ind: 8581

var am: 22 ind: 3510 3875 4140 4229 4329 4377 4501 4618 4717 4888 4978 5080 5129 5252 5363 5463 5545 5608 5687 5751 5820 5887

vera am: 1 ind: 1792

verdana am: 1 ind: 1775

violet am: 1 ind: 1522

visited am: 1 ind: 3306

vw am: 1 ind: 3246

we am: 1 ind: 8390

webkit am: 1 ind: 1666

weight am: 2 ind: 2533 6071

went am: 1 ind: 10882

when am: 3 ind: 8258 8322 8594

while am: 1 ind: 13540

white am: 1 ind: 3716

width am: 6 ind: 2389 2600 3038 3162 3234 3333

window am: 4 ind: 6617 6636 12287 14381

with am: 1 ind: 175

word am: 2 ind: 1890 1907

wrap am: 2 ind: 1895 3732

write am: 1 ind: 13090

wrong am: 1 ind: 10888

yellow am: 2 ind: 1400 5760

you am: 2 ind: 10028 13076

result of foundation:

while - am: 1 ind: 13540

a - am: 11 ind: 1315 1317 3290 3298 3471 10488 11523 12476 13092 13558 13595

**Вывод:** В ходе лабораторной работы разработали программное обеспечение для решения задачи с использованием стандартной библиотеки шаблонов в С++. Получили навыки использования классов контейнеров, итераторов, алгоритмов.

.