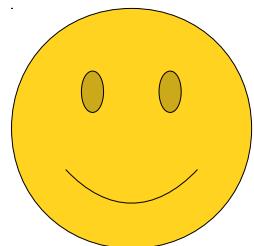
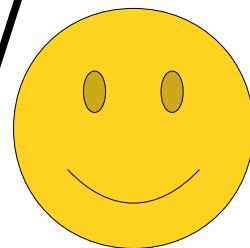


Assignment 0: Using the Debugger

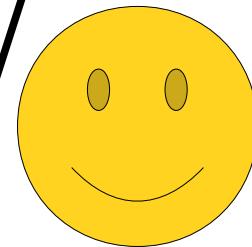


Hi everybody!

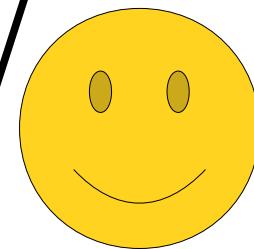
As part of Assignment 0, we'd like you to get a little bit of practice using the debugger in Qt Creator.



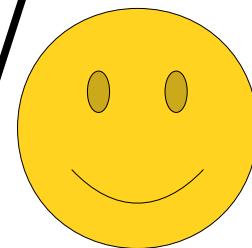
The debugger is a tool you can use to help see what your program is doing as you run it.



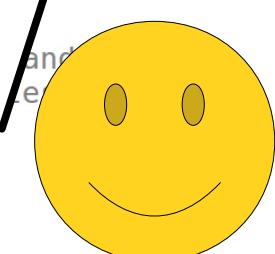
It's really useful for helping find errors in your programs, and the more practice you get with it, the easier it'll be to correct mistakes in the programs you write.



Think of this guide as a little tutorial walkthrough to help give you a sense of how to use the debugger and how to make sense of what you're seeing.



To start things off, open up the Name Hash program you ran in Part One of this assignment. Scroll down to the `nameHash` function so that you can see the entire function in your window.



```
40 * of the
41 *
42 * For tho
43 * treats
44 * It then
45 * F_p, wher
46 * some smaller prime numbers, a large prime, and
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime, and
51      * prime. These numbers were chosen because their product is less than
52      *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60      * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values of
64          * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp <Select Symbol>

name-hash
name.hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_n , where n is a large prime number, and evaluates that polynomial at
46 * $x = 1$. This is a very efficient way to hash names, though we won't cover this for CS106B,
47 * so we'll just implement the simple version.
48 *
49 * Time complexity: O(n), where n is the length of the name.
50 * Space complexity: O(1).
51 *
52 * This function is called from main() below.
53 *
54 * Parameters:
55 * first: The first character of the name.
56 * last: The last character of the name.
57 * kSmallPrime: A prime number used as a coefficient in the polynomial.
58 * kLargePrime: A larger prime number used as the modulus for the polynomial evaluation.
59 *
60 * Returns:
61 * The computed hash value, which is a number between 0 and 128.
62 *
63 * Note: The code assumes that the input name is non-empty and contains only ASCII characters.
64 *
65 * To use this function, call it like this:
66 * int hashVal = hashName(first, last);
67 *
68 * Example:
69 *> hashName('A', 'Z')
70 * 127
71 *

Move your mouse cursor so that it's in the space right before the line number for line 66.

Now, click the mouse!

int hashVal = 0;
/* Iterate across all the characters in the string name, updating the hash at each step.
 * lower-case letters are always less than 127.
for (char ch: first + last) {
 /* Convert the input character to lower case.
 * lower-case letters are always less than 127.
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

Issues Search Results Application Output Compile Output QML/JS Console General Messages



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp <Select Symbol>

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
Other files

40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case.
64 * lower-case letters are always less than 128.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70
71 }

When you do, you should see a red circle with a little hourglass pop up.

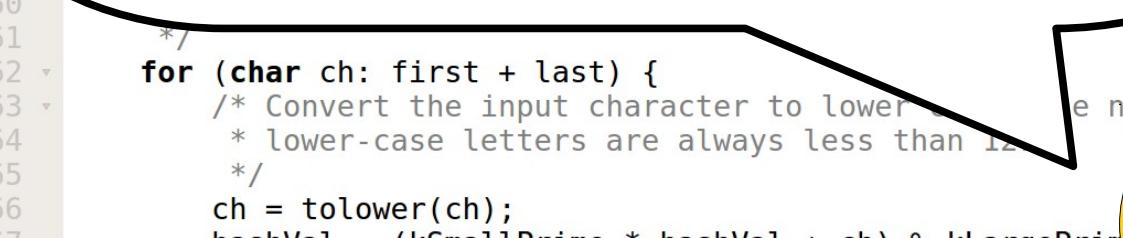
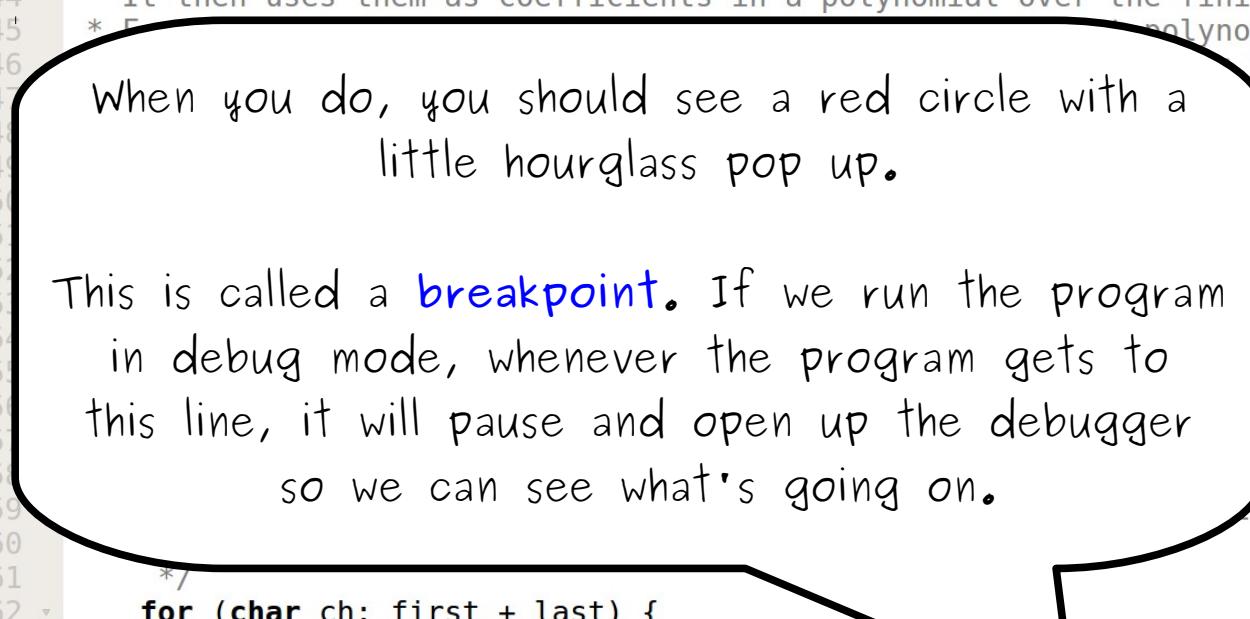
This is called a **breakpoint**. If we run the program in debug mode, whenever the program gets to this line, it will pause and open up the debugger so we can see what's going on.

Open Documents name_hash.cpp

name-hash
Debug

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp <Select Symbol>

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 * F_p , where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q . (You aren't expected to know this for CS106B,
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a small
51 less than
52
53 Now, we're going to run this program in debug
54 mode. To do so, click on the "run in debug mode"
55 button in the bottom-right corner of the screen.
56 It's the one just below the regular green "run"
57 button. When you do.
58
59 */
60
61 for (char ch: first + last) {
62 /* Convert the input character to lower case.
63 * Lower-case letters are always less than 128.
64 */
65 ch = tolower(ch);
66 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
67 }
68 return hashVal;
69 }
70
71

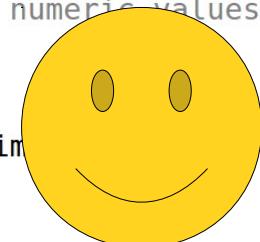
Open Documents name_hash.cpp

name-hash
Debug

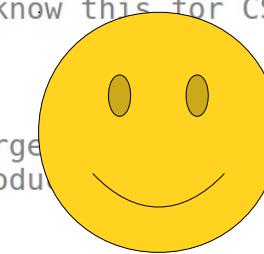
P- Type to locate (Ctrl...)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Now, we're going to run this program in debug mode. To do so, click on the "run in debug mode" button in the bottom-right corner of the screen. It's the one just below the regular green "run" button. When you do.



• you should see something like this! Notice that a bunch of extra panels popped up in Qt Creator. We'll talk about what each of these windows mean in a second.



```
* treats each character in the input name
* It then uses them as coefficients in a polynomial
* F_p, where p is a large prime number, and evaluates it at
* some smaller prime number q. (You aren't expected to know this for CS
* but we thought it might be fun!
*/
int nameHash() {
    /* ... */
    std::string name;
    int hashVal = 0;
    /* ... */
    for (char ch : name) {
        /* ... */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
}
```

The numeric values are 27.

Threads: Application started

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		

Type to locate (Ctrl+F)

Views

In the meantime, type in the first name Ada and hit enter, as shown here.

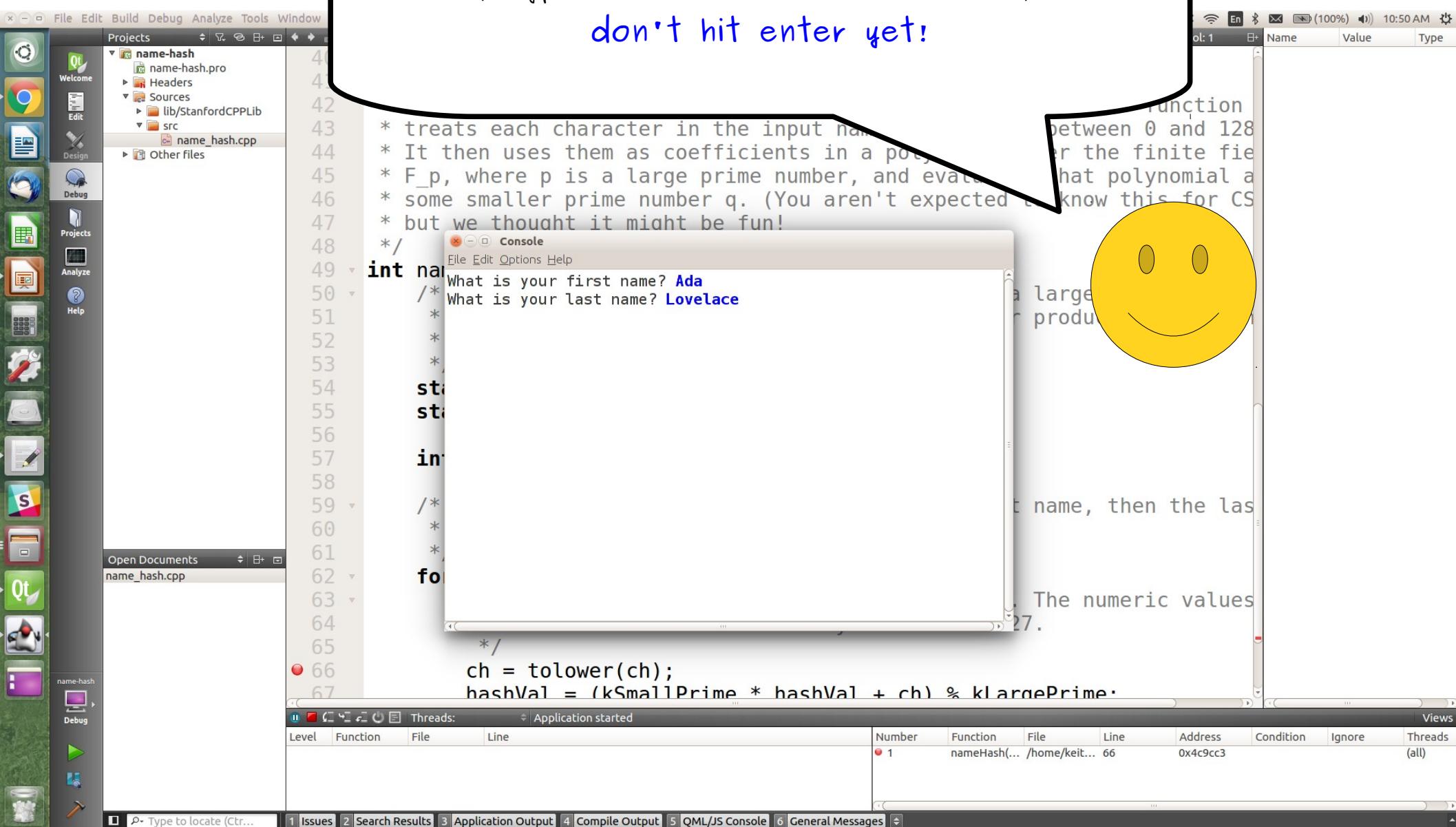
The screenshot shows the Qt Creator IDE interface. On the left is the project tree for "name-hash" with files like "name_hash.pro", "Headers", "Sources", "lib/StanfordCPPLib", and "src/name_hash.cpp". The main area shows the code for "name_hash.cpp". A callout bubble points to the console window, which displays the following text:

```
What is your first name? Ada
```

The code in "name_hash.cpp" includes a copyright notice and a license header. It defines a function "nameHash" that takes two strings and returns a long integer. The function uses a polynomial hash function based on prime numbers kSmallPrime and kLargePrime. It iterates through each character of the first name, converts it to lowercase, and adds its ASCII value multiplied by kSmallPrime to the current hash value, then takes the result modulo kLargePrime. The same process is repeated for the last name. The final hash value is returned.

The right side of the screen shows a large yellow smiley face icon. The status bar at the bottom indicates "Build: 1" and the system tray shows battery level, signal strength, and volume.

Now, type in "Lovelace" as a last name, but
don't hit enter yet!



The screenshot shows the Qt Creator IDE interface. On the left is the project tree for 'name-hash' containing files like 'name.hash.pro', 'Headers', 'Sources', 'lib/StanfordCPPLib', 'src', and 'Other files'. The main area shows the 'name_hash.cpp' file with C++ code for a polynomial hash function. A callout bubble points to the line where the user inputs their last name. A yellow smiley face icon is overlaid on the right side of the screen. The bottom of the window shows the Qt Creator status bar with tabs for 'Issues', 'Search Results', 'Application Output', 'Compile Output', 'QML/JS Console', and 'General Messages'.

```
* treats each character in the input name
* It then uses them as coefficients in a polynomial
* F_p, where p is a large prime number, and evaluates it at
* some smaller prime number q. (You aren't expected to know this for CS
* but we thought it might be fun!
*/
int nameHash(const QString& name) {
    /* ... */
    std::string strName = name.toStdString();
    int hashVal = 0;
    /* ... */
    for (int i = 0; i < strName.length(); ++i) {
        /* ... */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
}
```

What is your first name? Ada
What is your last name? Lovelace

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...)	/home/keit.../Qt/.../src/name_hash.cpp	66	0x4c9cc3	(all)		

As soon as you hit enter, a bunch of things are going to pop up in Qt Creator. Don't panic! It's normal.

The screenshot shows the Qt Creator IDE interface. On the left is the Projects panel with a project named "name-hash" selected. The main area is a code editor displaying "name_hash.cpp". The code implements a polynomial hash function for names. A callout bubble points to the terminal window, which shows the application's output:

```
What is your first name? Ada
What is your last name? Lovelace
```

A large yellow smiley face icon is overlaid on the right side of the screen. The status bar at the bottom shows "Threads: Application started".

Code snippet from "name_hash.cpp":

```
41
42
43     * treats each character in the input name
44     * It then uses them as coefficients in a polynomial
45     *  $F_p$ , where  $p$  is a large prime number, and evaluates it
46     * some smaller prime number  $q$ . (You aren't expected to know this for CS
47     * but we thought it might be fun!
48 */
49 int nameHash(const QString& name) {
50     /* ... */
51
52     std::string strName = name.toStdString();
53
54     int hashVal = 0;
55
56     /* ... */
57
58     for (int i = 0; i < strName.length(); ++i) {
59         /* ... */
60
61         ch = tolower(ch);
62         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
63
64     }
65
66
67 }
```

Output from the terminal window:

```
What is your first name? Ada
What is your last name? Lovelace
```

Qt Creator status bar:

```
Threads: Application started
```

Bottom navigation bar:

- Type to locate (Ctrl+F)
- Issues
- Search Results
- Application Output
- Compile Output
- QML/JS Console
- General Messages

Bottom right corner:

```
Vol: 1 Name Value Type
```

With that said, hit enter,
and watch the magic happen!

The screenshot shows the Qt Creator IDE interface. On the left is the project tree for 'name-hash' containing files like 'name.hash.pro', 'Headers', 'Sources', 'lib/StanfordCPPLib', 'src', and 'Other files'. The main code editor shows C++ code for generating a hash from names. A callout bubble points to the code block starting with 'int nameHash(const string& name)'.

```
* treats each character in the input name
* It then uses them as coefficients in a polynomial
* F_p, where p is a large prime number, and evaluates it
* some smaller prime number q. (You aren't expected to know this for CS
* but we thought it might be fun!
*/
int nameHash(const string& name) {
    /* ... */
    std::string first = name;
    std::string last = name;
    int hashVal = 0;
    /* ... */
    for (char ch : first) {
        /* ... */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    /* ... */
}
```

A callout bubble points to the line 'What is your first name? Ada' in the 'Console' window. A yellow smiley face icon is overlaid on the right side of the window. The 'Console' window also shows 'What is your last name? Lovelace'.

The 'Threads' tab at the bottom shows 'Application started' with one thread listed:

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
				1	nameHash(...)	/home/keit.../66	66	0x4c9cc3	(all)		

Shazam! We're back in Qt Creator, and there's tons of values showing up everywhere.

```
41
42
43
44
45
46
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * for lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```



Name	Value
_for_begin	@0x7fffffff030
_for_end	@0x7fffffff040
_for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120



There's a lot going on right here. Let's see what's happening.

The screenshot shows the Qt Creator IDE interface. On the left is the project navigation pane with a 'name-hash' project selected. The central area is a code editor displaying the following C++ code:

```
41
42
43
44
45
46
47
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64         * for lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

A callout bubble points from the handwritten text to the line 'return hashVal;' in the code. To the right of the code editor is a yellow smiley face icon. On the far right, there is a hex dump viewer showing memory addresses and values, and a threads tab showing the current thread state.

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

for_begin @0x7fffffff030
for_end @0x7fffffff040
for_range <not accessible>
ch 65 'A'
first @0x7fffffff100
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffff120

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * 2^31 - 1. This is important because the result of the hash function
53 * must fit into a 32-bit integer.
54 *
55 * The large prime is 16908799 and the small prime is 127. Both of
56 * these numbers are prime and have been checked with
57 * http://www.prime-numbers.net/.
58 *
59 * The algorithm for calculating the hash value is as follows:
60 *
61 * 1. Convert each character in the input strings to lowercase.
62 * 2. Initialize the hash value to 0.
63 * 3. For each character in the input strings, add its ASCII value to
64 * the current hash value. The ASCII value of a character is its
65 * numeric value. For example, the ASCII value of 'A' is 65.
66 * 4. Multiply the current hash value by the large prime (16908799).
67 * 5. Add the current hash value to the ASCII value of the character.
68 * 6. Repeat steps 4 and 5 for each character in the input strings.
69 * 7. The final hash value is the result.
70 *
71 * Note: The ASCII values of lowercase letters are always less than
72 * than the ASCII values of uppercase letters. For example, the ASCII
73 * value of 'a' is 97 and the ASCII value of 'A' is 65.

First, notice that our red breakpoint now has a yellow arrow in it.

then the last

case. The numeric values can be as high as 127.

for (char ch: first + last) {
 /* Convert the input character to lowercase. Lower-case letters are always less than
 * than the ASCII values of uppercase letters. For example, the ASCII value of 'a' is 97 and
 * the ASCII value of 'A' is 65.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch);
}
return hashVal;

Open Documents name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level Function File Line

0 nameHash name_hash... 66
1 Main name_hash... 31
2 main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

This yellow arrow indicates where in the program we are right now. The program stopped running at this line because we hit that breakpoint you set earlier.

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!)* /
```

```
int nameHash(string first, string last){    /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than 2^31 - 1. Prime 1
```

```
for (char ch: first + last) {    /* Convert the input character * lower-case letters are always less than uppercase letters */    ch = tolower(ch);    hashVal = (kSmallPrime * hashVal + ch);}
```

```
}
```

```
return hashVal;
```

```
}
```

then the last case. The numeric values for lowercase letters range from 97 to 127.

Open Documents
name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

name hash.pro Headers Sources lib/StanfordCPPLib src name_hash.cpp Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * 2^31 - 1. This means that the result of the multiplication will fit
53 * into a 32-bit integer. This is important for efficiency and memory
54 * usage.
55 * The large prime is 16908799 and the small prime is 127. Both of
56 * these numbers are prime and have been carefully chosen to make
57 * the resulting hash function as good as possible.
58 * The hash function is based on the formula:
59 * hashVal = (kSmallPrime * hashVal + ch) % kLargePrime
60 * where ch is the character being hashed. The hashVal starts at 0.
61 * The kSmallPrime is 127 and the kLargePrime is 16908799.
62 * The hash function is case-insensitive.
63 * The hash function is deterministic.
64 * The hash function is fast.
65 * The hash function is good.
66 * The hash function is good.
67 * The hash function is good.
68 * The hash function is good.
69 * The hash function is good.
70 * The hash function is good.
71 * The hash function is good.

for (char ch: first + last) {
 /* Convert the input character to lowercase. Lower-case letters are always less than
 * uppercase letters. This is important for consistency.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level Function File Line

0 nameHash name_hash... 66
1 Main name_hash... 31
2 main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Name Value

_for_begin @0x7fffffe030 std::for_each::begin
_for_end @0x7fffffe040 std::for_each::end
_for_range <not accessible> std::for_each::range
ch 65 'A' std::string::operator[](const std::string::size_type)
first 0 std::string::operator=(const std::string&)
hashVal 16908799 std::int<16908799>
kLargePrime 16908799 std::int<16908799>
kSmallPrime 127 std::int<127>
last @0x7fffffe120 std::string::operator[](const std::string::size_type)

then the last

case. The numeric values range from 0 to 127.

me;

Whenever you pop up the debugger, it's good to figure out exactly where you are in the program that you're running, so you'll get into the habit of checking for this yellow arrow.

The screenshot shows the Qt Creator IDE interface. On the left is the project navigation sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main window displays the code for `nameHash.cpp`. The code defines a function `nameHash` that takes two strings and returns an integer. It uses two prime numbers, `kLargePrime` and `kSmallPrime`, to calculate a hash value by iterating through each character of the input names. A callout bubble points from the text "This is called the call stack." to the call stack panel at the bottom.

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!)* /  
int nameHash(string first, string last){  
    /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than  $2^{31} - kLargePrime - 1$ . */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first name, then the last name, updating the hash at each step. */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case. The numbers are lowercase. */  
        ch = tolower(ch);  
        hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;  
    }  
    return hashVal;  
}
```

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash... 66	
1	Main	name_hash... 31	
2	Main	main.cpp 23	
3	startupMain	platform.cpp 2208	
4	main	name hash... 27	

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keith...	66	0x4c9cc3			(all)

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

for_begin @0x7fffffff030
for_end @0x7fffffff040
for_range <not accessible>
ch 65 'A'
first 0
hashVal 16908799
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffff120

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54 static const int kLargePrime = 16908799;
55 static const int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numbers
64 * lower case
65 */
66 ch = hashVal;
67 }
68 return hashVal;
69 }
70
71 }

Open Documents name_hash.cpp

name-hash

Debug

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keith...	66	0x4c9cc3			(all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

The screenshot shows the Qt Creator IDE interface. On the left is the project navigation sidebar with icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main area displays the code for `nameHash.cpp`. The code implements a polynomial hash function:`45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * 2^31 - kLargePrime - 1.
53 */
54 static const int kLargePrime = 16908799;
55 static const int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numbers
64 * lower case
65 */
66 ch = tolower(ch);
67 hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }

A yellow smiley face icon is positioned on the right side of the code editor. A black callout bubble originates from the smiley face and points to the handwritten note below.

However, the yellow arrow can't tell us exactly how we got to this part of the program. What part of the program actually called nameHash?

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keith...	66	0x4c9cc3			(all)

The call stack can tell us exactly that!

```
name hash.cpp
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numbers
64         * lower case.
65         */
66         ch = tolower(ch);
67         hashVal = (hashVal * kSmallPrime + ch) % kLargePrime;
68     }
69
70     return hashVal;
71 }
```

Level	Function	File	Line
0	nameHash	name_hash.cpp	66
1	Main	name_hash.cpp	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...)	/home/keith/...	66	0x4c9cc3			(all)

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030 std::for_each::begin

_for_end @0x7fffffff040 std::for_each::end

_for_range <not accessible> std::for_each::range

ch 65 'A' std::char_traits<char>

first 0 std::vector<char>

hashVal 16908799 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff0120 std::vector<char>::end

Welcome Edit Design Debug Projects Analyze Help

Open Documents name_hash.cpp

name-hash

name-hash.pro

Headers

Sources

lib/StanfordCPPLib

src

name_hash.cpp

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numbers are lowercase.

59 */

60 **ch =**

61 **hashVal**;

62 }

63 **return** hashVal;

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level Function File Line

0 nameHash name_hash... 66

1 Main name_hash... 31

2 main main.cpp 23

3 startupMain platform.cpp 2208

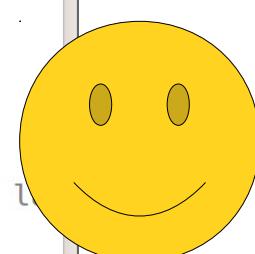
4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keith/... 66 0x4c9cc3 (all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages



Notice that the call stack lists a series of different functions in order. Here, it has `nameHash` (where we are now) at the top, and right below that is `Main`.

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

name hash.pro Headers Sources lib/StanfordCPPLib src name_hash.cpp Other files

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030 std::string::begin

_for_end @0x7fffffff040 std::string::end

_for_range <not accessible> std::string::range

ch 65 'A'

first 0 std::string::first

hashVal 16908799 std::int32_t::operator=

kLargePrime 16908799 std::int32_t::operator=

kSmallPrime 127 std::int32_t::operator=

last @0x7fffffff0120 std::string::last

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47

48

49 **int** nameHash(**string** first, **string** last){

50 /* This hashing scheme needs two prime numbers, a large prime and a

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 **static const int** kLargePrime = 16908799;

55 **static const int** kSmallPrime = 127;

56

57 **int** hashVal = 0;

58

59 /* Iterate across all the characters in the first name, then the last

60 * name, updating the hash at each step.

61 */

62 **for (char ch: first + last) {**

63 /* Convert the input character to lower case. The numbers are

64 * lower case here.

65 */

66 **ch** = **hashVal** = **hashVal** % **kLargePrime**;

67 }

68 }

69 } **return** hashVal;

70 }

71 }

66

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level Function File Line

0 nameHash name_hash.cpp 66

1 Main main.cpp 23

2 Main main.cpp 23

3 startupMain platform.cpp 2208

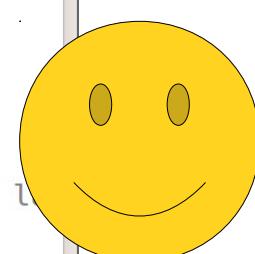
4 main name_hash.cpp 27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages



Go and double-click the call to Main on Level 1.
When you do.

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp # Line: 31, Col: 5

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual
38 * to talk more
39 * the meaning
40 * of the input
41 *
42 * For those
43 * treats each
44 * It then uses
45 * F n where n is
```

Open Documents name_hash.cpp

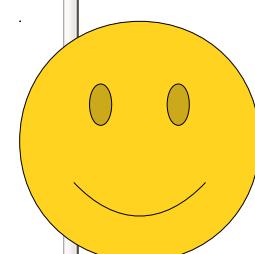
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	766504679	int

Views

Issues Search Results Application Output Compile Output QML/JS Console General Messages



• you'll end up over here!

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp # Line: 31, Col: 5

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual
38 * to talk more
39 * the meaning
40 * of the input
41 *
42 * For those
43 * treats each
44 * It then uses
45 * F n where n is
```

Open Documents name_hash.cpp

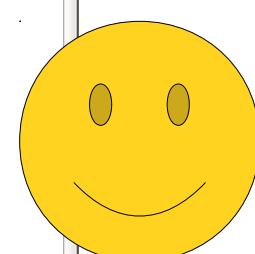
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Name Value Type

first @0x7fffffff0a0 std::string
last @0x7fffffff0c0 std::string
hashValue 766504679 int

Notice that the highlighted line here includes a call to the `nameHash` function. This is the part of the code that actually called `nameHash`, which is how we got to the line with the breakpoint!



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp # Line: 31, Col: 5

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual
38 * to talk more
39 * the meaning
40 * of the input
41 *
42 * For those
43 * treats each
44 * It then uses
45 * F n where n is
```

Open Documents name_hash.cpp

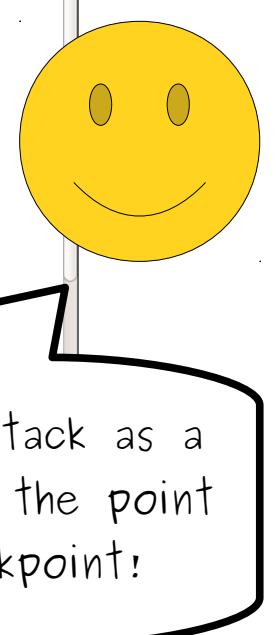
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Name	Value	Type
first	@0x7fffffff0e00	std::string
last	@0x7fffffff0e00	std::string
hashValue	766504679	int

Views

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



Generally speaking, you can use the call stack as a way to see which function calls got us to the point where the program paused at the breakpoint!

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp # Line: 31, Col: 5

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
Other files

```
#include "console.h"
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the actual
 * to talk more
 * the meaning
 * of the input
 *
 * For those
 * treats each
 * It then uses
 * Finally where n
```

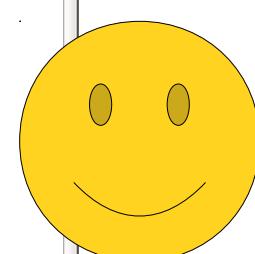
Open Documents name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	766504679	int

You might notice that there's some more stuff in the call stack beyond just main and nameHash. What are those?



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp # Line: 31, Col: 5

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
Other files

```
18 #include "console.h"
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets us use the function
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name? ");
29     string last = getLine("What is your last name? ");
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashValue << endl;
34     return 0;
35 }
36
37 /* This is the actual
38 * to talk more
39 * the meaning
40 * of the input
41 *
42 * For those
43 * treats each
44 * It then uses
45 * F n where n is
```

Open Documents name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

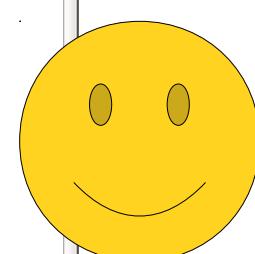
Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	1
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Views

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages



Let's find out! Double-click on the line marked "Main" on Level 2. When you do...

File Edit Build Debug Analyze Tools Window Help

Projects main.cpp Main(int, char *[]) int # Line: 23, Col: 5 Name Value Type

Welcome Edit Design Debug Projects Analyze Help

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

main.cpp
name_hash.cpp

1 /* ... */
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

#include <iostream>

#ifndef SPL_AUTOGRADER_MODE
int Main(int, char* /*argv*/[]) {
 extern int Main();
 return Main();
}
#endif // SPL_AUTOGRADER_MODE

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level Function File Line Number Function File Line Address Condition Ignore Threads

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

... you'll end up with something that looks like this.

A yellow smiley face icon with a speech bubble pointing towards the text '... you'll end up with something that looks like this.' A large black line drawing of a speech bubble surrounds the text.

File Edit Build Debug Analyze Tools Window Help

Projects main.cpp Main(int, char *[]) int # Line: 23, Col: 5 Name Value Type

Welcome Edit Design Debug Projects Analyze Help

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

1 /* ... */
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

#include <iostream>

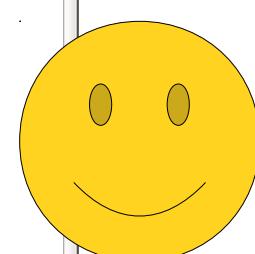
#ifndef SPL_AUTOGRADER_MODE
int Main(int, char* /*argv*/[]) {
 extern int Main();
 return Main();
}
#endif // SPL_AUTOGRADER_MODE

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



Yikes! This looks Hairy and Scary! What happened?

File Edit Build Debug Analyze Tools Window Help

Projects main.cpp Main(int, char *[]) int # Line: 23, Col: 5 Name Value Type

Welcome Edit Design Debug Projects Analyze Help

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

1 /* ... */
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

#include <iostream>

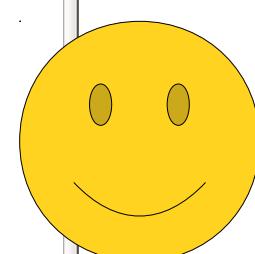
#ifndef SPL_AUTOGRADER_MODE
int Main(int, char* /*argv*/[]) {
 extern int Main();
 return Main();
}
#endif // SPL_AUTOGRADER_MODE

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

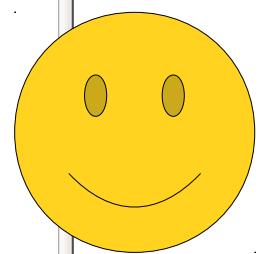


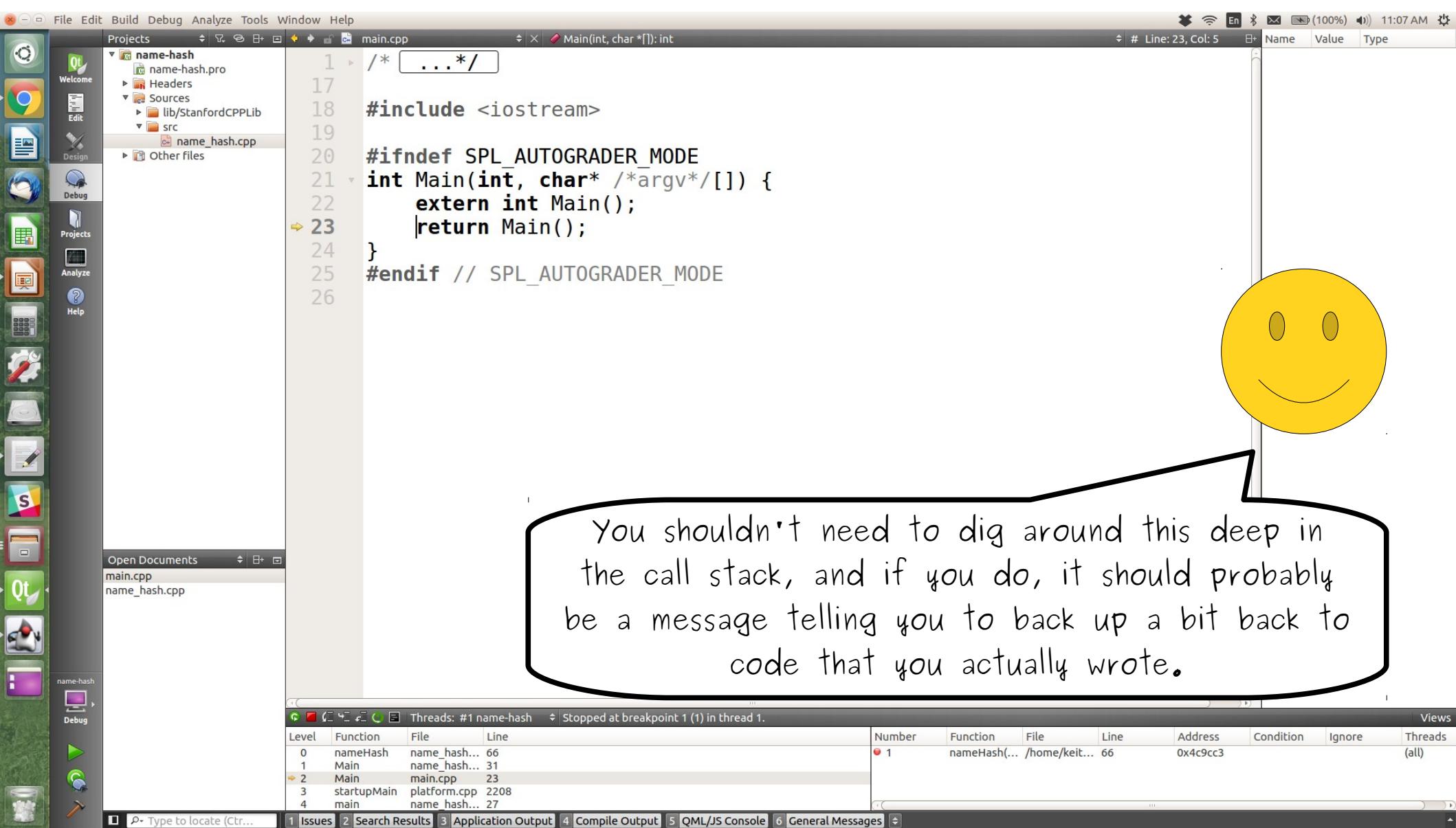
Whenever you start up a program in CS106B, there's a little bit of code that we automatically call for you, which does things like setting up the console.

A screenshot of the Qt Creator IDE interface. The top menu bar includes File, Edit, Build, Debug, Analyze, Tools, Window, Help, and a system tray with battery level (100%) and time (11:07 AM). The left sidebar contains icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help. The main window shows a project named "name-hash" with files main.cpp and name_hash.cpp. The code editor displays the following C++ code:

```
1  /* ... */
17
18
19
20 #ifndef SPL_AUTOGRADER_MODE
21 int Main(int, char* /*argv*/[]) {
22     extern int Main();
23     return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26
```

An annotation is overlaid on the right side of the screen. It features a large yellow smiley face at the top right. A black line connects the smiley face to a rounded rectangular callout box in the center-right area. Inside the callout box, the text "This code will show up in the call stack below your actual program." is written in a casual, handwritten-style font.





The screenshot shows the Qt Creator IDE interface. The top menu bar includes File, Edit, Build, Debug, Analyze, Tools, Window, Help, and a system tray with icons for battery, signal, and volume.

The left sidebar contains icons for Welcome, Edit, Design, Debug, Projects, Analyze, and Help, with 'Projects' currently selected. The main window displays a code editor with the file `main.cpp` open. The code includes a header guard, an `#include <iostream>`, and a function `Main` with a specific implementation for SPL_AUTOGRADER_MODE.

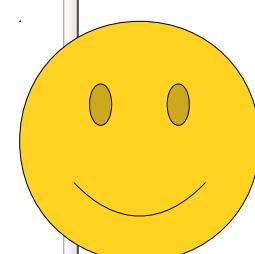
The bottom part of the interface shows the Threads view, which is currently stopped at breakpoint 1 in thread 1. The call stack shows the following levels:

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

The status bar at the bottom indicates the current line and column: Line: 23, Col: 5. The bottom navigation bar includes tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.

A large, hand-drawn style callout bubble with a yellow smiley face points from the right side of the screen towards the center, containing the following text:

You shouldn't need to dig around this deep in the call stack, and if you do, it should probably be a message telling you to back up a bit back to code that you actually wrote.



So let's jump back to the code that we actually wrote.

```
/* ... */

#include <iostream>

#ifndef SPL_AUTOGRADER_MODE
int Main(int, char* /*argv*/[]) {
    extern int Main();
    return Main();
}
#endif // SPL_AUTOGRADER_MODE
```

File Edit Build Debug Analyze Tools Window Help

Projects main.cpp Main(int, char *[]) int

Line: 23, Col: 5 Name Value Type

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

1 /* ... */
17
18
19
20
21 int Main(int, char* /*argv*/[]){
22 extern int Main();
23 return Main();
24 }
25 #endif // SPL_AUTOGRADER_MODE
26

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

To do that, double-click on Level 0, the call to nameHash. When you do.

Level	Function	File	Line
0	nameHash	name hash...	66
1	Main	name hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030 std::for_each::iterator<std::vector<char> const&_for_end @0x7fffffff040 std::for_each::iterator<char> <not accessible> _for_range ch 65 'A' first hashVal 0 kLargePrime 16908799 kSmallPrime 127 last @0x7fffffff000

name hash.hash
name hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numbers are lowercase.

59 */

60 **ch** = **hashVal**;

61 }

62 **return** hashVal;

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level Function File Line

0 nameHash name_hash... 66

1 Main name_hash... 31

2 Main main.cpp 23

3 startupMain platform.cpp 2208

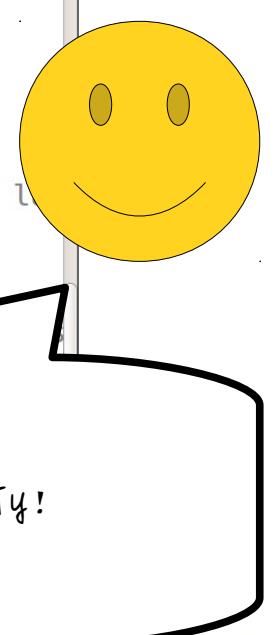
4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages



You'll be teleported back to safety!

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030
_for_end @0x7fffffff040
_for_range <not accessible>
ch 65 'A'
first @0x7fffffff100
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffff120

```
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal
58
59     /* Iterates over each character in the string
60     * naming the character ch
61     */
62     for (char ch : last) {
63         /*
64         *
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70
71 }
```

Let's quickly recap what we've seen so far.



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...)	/home/keit...	66	0x4c9cc3			(all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

To set a breakpoint so that we can pause the program and look around, click in the margin just before the line number where you want to pause.

The screenshot shows the Qt Creator IDE interface. On the left is a vertical toolbar with various icons for file operations like Open, Save, and Find. The main window has a title bar with tabs for "name hash.cpp" and "nameHash(string, string): int". The code editor displays a C++ function named `nameHash` with annotations explaining its purpose. A callout bubble points from the text "To set a breakpoint..." to the line number 66 in the margin, which is highlighted with a red arrow. To the right of the callout is a large yellow smiley face icon. The bottom of the screen shows the debugger's stack trace and registers tables.

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!
*/
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than  $2^{31} - kLargePrime - 1$ .
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numerical lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

The screenshot shows the Qt Creator IDE interface. The main window displays a C++ file named `name_hash.cpp` with the following code:

```
* F_p, where p is a large prime number, and evaluates that polynomial at
* some smaller prime number q. (You aren't expected to know this for CS
* but we thought it might be fun!
*/
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a
     * prime. These numbers were chosen because their product is less than
     * 2^31 - kLargePrime - 1.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numerical
         * lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

A large callout bubble is overlaid on the code, containing the handwritten note: "Once the breakpoint is reached, it will pull up all sorts of useful information." A yellow smiley face icon is positioned to the right of the callout.

The status bar at the bottom indicates: "Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1."

The yellow arrow points out where we are right now.

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!
*/
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a
     * prime. These numbers were chosen because their product is less than
     * 2^31 - kLargePrime - 1.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numerical
         * lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

name hash.cpp # Line: 66, Col: 9

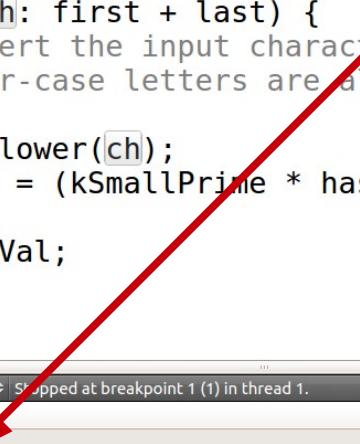
Name	Value
_for_begin	@0x7fffffff030
_for_end	@0x7fffffff040
_for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

The call stack shows us how we got into the current function.



A yellow smiley face icon is located in the bottom right corner of the IDE window.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Name	Value
_for_begin	@0x7fffffff030
_for_end	@0x7fffffff040
_for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp # Line: 66, Col: 9

name hash.pro
Headers
Sources
lib/StanfordCPPLib
src
Other files

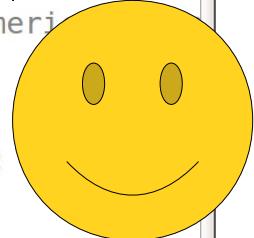
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54
55
56
57
58
59
60
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Now, let's see how we can read the values of the variables in this function.



Name Value

Name	Value
_for_begin	@0xfffffffffe030
_for_end	@0xfffffffffe040
_for_range	<not accessible>
ch	65 'A'
first	@0xfffffffffe100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0xfffffffffe120

Look up at this panel over here.



```
45
46
47 * but we o be run:
48 */
49
50     nameHash(string first, string last){
51
52     /* This hashing scheme needs two prime numbers, a large prime and a
53     prime. These numbers were chosen because their product is less than
54     2^31 - kLargePrime - 1.
55
56     static const int kLargePrime = 16908799;
57     static const int kSmallPrime = 127;
58
59     int hashVal = 0;
60
61     /* Iterate across all the characters in the first name, then the last
62     * name, updating the hash at each step.
63     */
64
65     for (char ch: first + last) {
66         /* Convert the input character to lower case. The numeric values
67         * lower-case letters are always less than 127.
68         */
69         ch = tolower(ch);
70         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
71     }
72     return hashVal;
73 }
```

The screenshot shows a Qt Creator IDE interface. A callout bubble points from the handwritten text "Look up at this panel over here." to the bottom right corner of the window, which contains the "Registers" panel. This panel displays memory dump information with columns for Name, Value, and Type. A yellow smiley face icon is overlaid on the left side of the code editor area. The code editor shows C++ code for a name hashing function, including comments about prime number selection and a loop that iterates over the characters of a combined first and last name, applying a rolling hash algorithm using two primes: 16908799 and 127. The Registers panel lists several memory locations, including pointers to arrays and individual characters, along with their addresses and values.

This window lets you take a look at all the values of the local variables that are in scope right now.



Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.				
l	Function	File	Line	
	nameHash	name_hash... 66		
	Main	name_hash... 31		
	Main	main.cpp 23		
	startupMain	platform.cpp 2208		
	main	name_hash... 27		

Depending on what OS you're using, these might be in a different order, and there might be some weird-looking ones in there in addition to nicer ones like ch and hashVal.



```
45
46
47 * but we o
48 */
49
50     nameHash(string first, string last){
51         /* This hashing scheme needs two prime numbers, a large prime and a
52          prime. These numbers were chosen because their product is less than
53          2^31 - kLargePrime - 1.
54
55         static const int kLargePrime = 16908799;
56         static const int kSmallPrime = 127;
57
58         int hashVal = 0;
59
60         /* Iterate across all the characters in the first name, then the last
61          name, updating the hash at each step.
62         */
63         for (char ch: first + last) {
64             /* Convert the input character to lower case. The numeric values
65              for lower-case letters are always less than 127.
66             */
67             ch = tolower(ch);
68             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69         }
70         return hashVal;
71     }
```

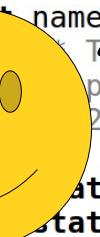
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

If we ignore the weird-looking ones, we can see some nice, familiar names.





```
45
46
47 * but we
48 */
49 int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a
     * prime. These numbers were chosen because their product is less than
     * 2^31 - kLargePrime - 1.
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

    /* Iterate across all the characters in the first name, then the last
     * name, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric values
         * lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

Level	Function	File	Line	Number	Function	File	Line	Address
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9c3
1	Main	name_hash...	31					
2	Main	main.cpp	23					
3	startupMain	platform.cpp	2208					
4	main	name_hash...	27					

For example, here you can see the values of kLargePrime and kSmallPrime, which match the values they were declared with.



```
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

* but we can't be run:
*/
+ nameHash(string first, string last){
 /* This hashing scheme needs two prime numbers, a large prime and a
 prime. These numbers were chosen because their product is less than
 $2^{31} - \text{kLargePrime} - 1$.

 static const int kLargePrime = 16908799;
 static const int kSmallPrime = 127;

 int hashVal = 0;

 /* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
 for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values
 * lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
 }
 return hashVal;

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

We can also see that, at this point, hashVal is still zero.



A yellow smiley face icon is positioned on the left side of the code editor area, with a blue arrow pointing from it towards the variable declaration `int hashVal = 0;`.

```
45
46
47 * but we do not need to be run:
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51      * prime. These numbers were chosen because their product is less than
52      * 2^31 - kLargePrime - 1.
53
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60      * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values
64          * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

The code is part of a C++ program named `nameHash`. It defines a function `nameHash` that takes two strings, `first` and `last`, and returns their combined hash value. The function uses two prime numbers, `kLargePrime` and `kSmallPrime`, to calculate the hash. The variable `hashVal` is initialized to 0. The function iterates over all characters in both strings, converting them to lowercase using `tolower` and updating the `hashVal` by applying the formula $(kSmallPrime * hashVal + ch) \% kLargePrime$.

On the right side of the interface, there is a debugger window showing the current state of variables:

Name	Value
_for_begin	@0x7fffffff030
_for_end	@0x7fffffff040
_for_range	<not accessible>
ch	65 'A'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

The variable `hashVal` is currently 0, which corresponds to the handwritten note in the speech bubble.

As we walk through the program one step at a time,
we'll see these values change.



```
45
46
47 * but we do not have to be run:
48 */
49
    nameHash(string first, string last){
        /* This hashing scheme needs two prime numbers, a large prime and a
         * prime. These numbers were chosen because their product is less than
         *  $2^{31} - kLargePrime - 1$ .
50
        static const int kLargePrime = 16908799;
51
        static const int kSmallPrime = 127;
52
        int hashVal = 0;
53
        /* Iterate across all the characters in the first name, then the last
         * name, updating the hash at each step.
         */
54
        for (char ch: first + last) {
            /* Convert the input character to lower case. The numeric values
             * for lower-case letters are always less than 127.
             */
55
            ch = tolower(ch);
            hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
        }
        return hashVal;
    }
}
```

The screenshot shows a Qt Creator IDE interface. The central area displays a C++ code editor with the file 'name_hash.cpp' open. A callout bubble from the top right points to the variable 'ch' in the code, which is highlighted with a yellow box. To the right of the code editor is a 'Registers' tool window showing memory dump values. At the bottom of the screen is a navigation bar with tabs for 'Issues', 'Search Results', 'Application Output', 'Compile Output', 'QML/JS Console', and 'General Messages'. On the far left, there's a vertical toolbar with various icons for file operations like 'Welcome', 'Edit', 'Design', 'Debug', 'Projects', 'Analyze', and 'Help'.

Now, let's take a look at this for loop.



```
45
46
47 * but we do not have to be run:
48 */
49
50     nameHash(string first, string last){
51         /* This hashing scheme needs two prime numbers, a large prime and a
52          prime. These numbers were chosen because their product is less than
53          2^31 - kLargePrime - 1.
54
55         static const int kLargePrime = 16908799;
56         static const int kSmallPrime = 127;
57
58         int hashVal = 0;
59
60         /* Iterate across all the characters in the first name, then the last
61          name, updating the hash at each step.
62         */
63
64         for (char ch: first + last) {
65             /* Convert the input character to lower case. The numeric values
66              of lower-case letters are always less than 127.
67             */
68             ch = tolower(ch);
69             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
70         }
71     return hashVal;
72 }
```

The code snippet shows a C++ function named `nameHash` that takes two strings, `first` and `last`, as parameters. It uses two prime numbers, `kLargePrime` and `kSmallPrime`, to calculate a hash value. The function iterates over all characters in both strings, converting them to lowercase using `tolower`, and updates the hash value by multiplying it with `kSmallPrime` and adding the character value. A yellow smiley face icon is overlaid on the code editor window. A callout bubble from the top right points to the `for` loop. A dashed blue rectangle highlights the loop body.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Number	Function	File	Line	Address	Condition	Ignore	Threads
1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

This loop is a **range-based for loop**. It says "for each character in the string `first + last`, do something with that character."



```
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

* but we can't be run:
*/
+ nameHash(string first, string last){
 /* This hashing scheme needs two prime numbers, a large prime and a
 prime. These numbers were chosen because their product is less than
 $2^{31} - kLargePrime - 1$.

 static const int kLargePrime = 16908799;
 static const int kSmallPrime = 127;

 int hashVal = 0;

 /* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
 for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values
 * lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
 }
 return hashVal;

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	66	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

Remember (from a while back) that we entered the name Ada Lovelace.



```
45
46
47 * but we can't be run:
48 */
49
50     nameHash(string first, string last){
51         /* This hashing scheme needs two prime numbers, a large prime and a
52          prime. These numbers were chosen because their product is less than
53          2^31 - kLargePrime - 1.
54
55         static const int kLargePrime = 16908799;
56         static const int kSmallPrime = 127;
57
58         int hashVal = 0;
59
60         /* Iterate across all the characters in the first name, then the last
61          name, updating the hash at each step.
62
63         */
64
65         for (char ch: first + last) {
66             /* Convert the input character to lower case. The numeric values
67              of lower-case letters are always less than 127.
68              */
69             ch = tolower(ch);
70             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
71         }
72         return hashVal;
73     }
74 }
```

The screenshot shows a Qt Creator IDE interface. On the left is the project tree for 'name-hash' containing files like 'name_hash.pro', 'Headers', 'Sources', 'lib/StanfordCPPLib', and 'src/name_hash.cpp'. The main editor window displays the 'name_hash.cpp' code. A callout bubble from the top text points to the 'nameHash' function. A dashed blue rectangle highlights the loop that iterates over the characters of the input names, calculating a rolling hash. The bottom status bar shows the message 'Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.' The bottom navigation bar includes tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.

If we take a look at the current value of the variable `ch`, we can see that it has the value A. That's the first letter of the name Ada Lovelace.



```
File Edit Build Debug Analyze Tools Window Help
Projects
  name-hash
    name-hash.pro
    Headers
    Sources
      lib/StanfordCPPLib
      src
        name_hash.cpp
    Other files
  nameHash.h
  main.cpp
  name_hash.cpp
  Open Documents
    main.cpp
    name_hash.cpp
  name-hash
    Debug
Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.
Level Function File Line
0 nameHash name_hash... 66
1 Main name_hash... 31
2 Main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27
Number Function File Line Address Condition Ignore Threads
1 nameHash(... /home/keit... 66 0x4c9cc3 (all)
Name Value
for_begin @0x7fffffe030
for_end @0x7fffffe040
for_range not accessible
ch 65 'A'
first @0x7fffffe100
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffe120

```

* but we can't see the value of `ch` because it's not accessible.
*/
 * This hashing scheme needs two prime numbers, a large prime and a
 prime. These numbers were chosen because their product is less than
 $2^{31} - kLargePrime - 1$.

 static const int kLargePrime = 16908799;
 static const int kSmallPrime = 127;

 int hashVal = 0;

 /* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
 for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values
 * for lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
 }
 return hashVal;

So now we know where we are (line 66), how we got there (main called nameHash), and the values in the program at this point.



```
File Edit Build Debug Analyze Tools Window Help
Projects name-hash
  - name-hash.pro
  - Headers
  - Sources
    - lib/StanfordCPPLib
    - src
      - name_hash.cpp
  - Other files
Welcome Edit Design Debug Projects Analyze Help
En (100%) 11:10 AM
Name Value
▶ _for_begin @0x7fffffff030 std::string::begin
▶ _for_end @0x7fffffff040 std::string::end
▶ _for_range <not accessible> std::string::range
ch 65 'A' char
▶ first @0x7fffffff100 std::string::first
hashVal 0 int
kLargePrime 16908799 int
kSmallPrime 127 int
last @0x7fffffff120 std::string::last
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
* but we can't be run:
*/
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a
     * smaller prime. These numbers were chosen because their product is less than
     * 2^31 - kLargePrime - 1.
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

    /* Iterate across all the characters in the first name, then the last
     * name, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric values
         * for lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.
Level Function File Line
0 nameHash name_hash... 66
1 Main name_hash... 31
2 Main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27
Number Function File Line Address Condition Ignore Threads
1 nameHash(... /home/keit... 66 0x4c9cc3 (all)
Views
Issues Search Results Application Output Compile Output QML/JS Console General Messages
```

Now, let's do something really cool - we're going to run this program one line at a time, watching what happens at each step!



```
45
46
47 * but we can't be run:
48 */
49
50     nameHash(string first, string last){
51         /* This hashing scheme needs two prime numbers, a large prime and a
52          prime. These numbers were chosen because their product is less than
53          2^31 - kLargePrime - 1.
54
55         static const int kLargePrime = 16908799;
56         static const int kSmallPrime = 127;
57
58         int hashVal = 0;
59
60         /* Iterate across all the characters in the first name, then the last
61          name, updating the hash at each step.
62         */
63         for (char ch: first + last) {
64             /* Convert the input character to lower case. The numeric values
65              for lower-case letters are always less than 127.
66             */
67             ch = tolower(ch);
68             hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
69         }
70         return hashVal;
71     }
```

The screenshot shows a Qt Creator IDE interface. The central area displays a C++ code editor with the file 'name_hash.cpp' open. A callout bubble from the top right points to the code line '/* but we can't be run: */'. A yellow smiley face icon is overlaid on the left side of the code editor. The bottom right corner features a debugger window showing a list of threads and a variable table with columns for Name, Value, Type, etc. The status bar at the bottom indicates 'Threads: #1 name-hash' and 'Stopped at breakpoint 1 (1) in thread 1.'

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 65 'A' std::string::iterator

first @0x7fffffff100 std::string::iterator

hashVal 0 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::string::iterator

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 **ch** = tolower(**ch**);

61 hashVal = (kSmallPrime * hashVal + **ch**) % kLargePrime;

62 }

63

64 **return** hashVal;

65 }

66 }

67

68

69

70

71 }

Open Documents main.cpp name_hash.cpp

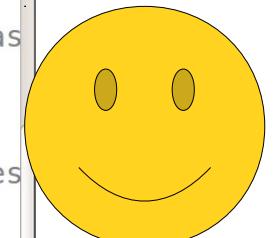
Threads: #1 name-hash

Level	Function	File	Line
0	nameHash	name_hash... 66	
1	Main	name_hash... 31	
2	Main	main.cpp 23	
3	startupMain	platform.cpp 2208	
4	main	name hash... 27	

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

Right above the stack trace, you'll see there are some small button icons.



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

```

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 * some smaller prime number q. (You aren't expected to know this for CS
47 * but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values for
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 
```

Open Documents main.cpp name_hash.cpp

name hash

Threads: #1 name-hash

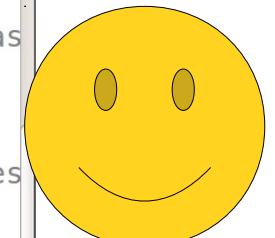
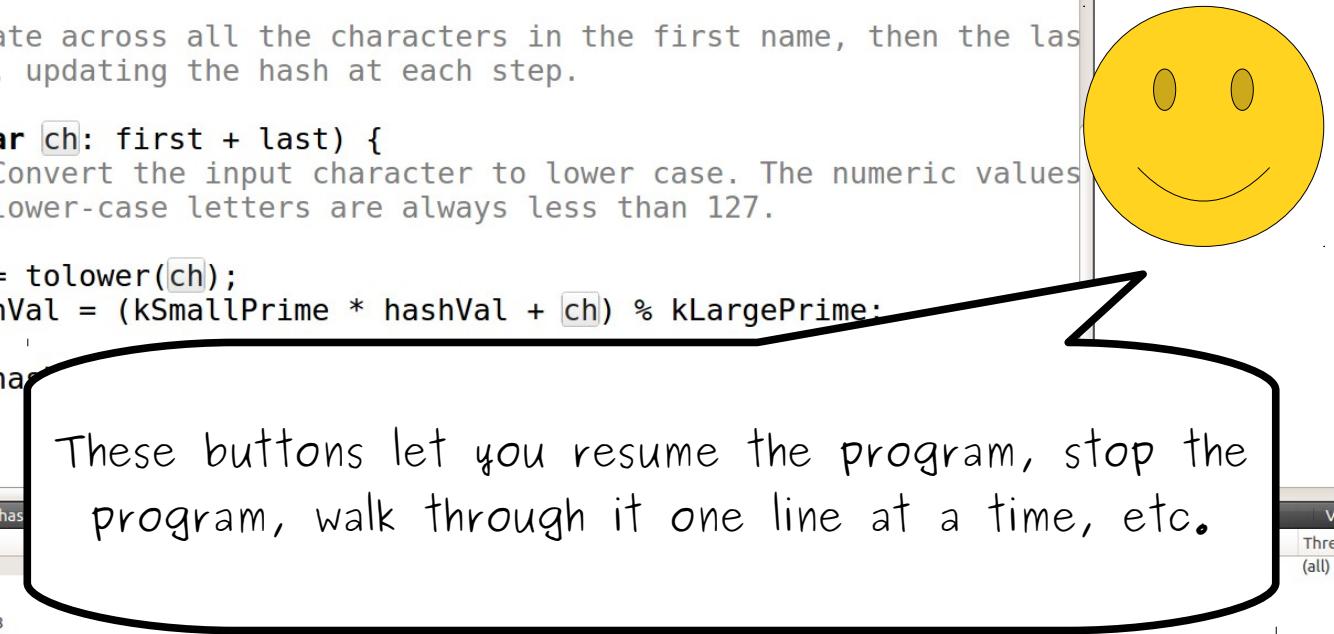
Level	Function	File	Line
0	nameHash	name_hash...	66
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

Line: 66, Col: 9

Name Value Type

- _for_begin @0x7fffffff030 std::string
- _for_end @0x7fffffff040 std::string
- _for_range <not accessible> std::string
- ch 65 'A' std::string
- first @0x7fffffff100 std::string
- hashVal 0 int
- kLargePrime 16908799 int
- kSmallPrime 127 int
- last @0x7fffffff120 std::string

These buttons let you resume the program, stop the program, walk through it one line at a time, etc.

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030 std::for_each::begin

_for_end @0x7fffffff040 std::for_each::end

_for_range <not accessible> std::for_each::range

ch 65 'A' std::char_traits<char>

first @0x7fffffff100 std::vector<char>::begin

hashVal 0 std::vector<char>::value_type

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::vector<char>::end

Welcome Edit Design Debug Projects Analyze Help

Open Documents main.cpp name_hash.cpp

name-hash

name hash.pro

Headers

Sources

lib/StanfordCPPLib

src

name_hash.cpp

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a small prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = tolower(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 }

64 **return** hashVal;

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Threads: #1 name-has

Level Function File Line

0 nameHash name_hash... 66

1 Main name_hash... 31

2 Main main.cpp 23

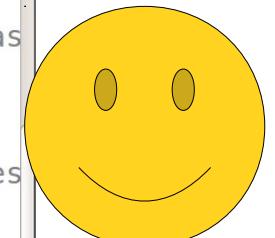
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

Move your mouse so that you're hovering over the button that's third from the left. If you hover over it, it should say "step over."



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 66, Col: 9

Name Value

_for_begin @0x7fffffff030 std::for_each::begin

_for_end @0x7fffffff040 std::for_each::end

_for_range <not accessible> std::for_each::range

ch 65 'A'

first @0x7fffffff100 std::vector<char>::begin

hashVal 0 std::vector<char>::value_type

kLargePrime 16908799 std::vector<char>::size_type

kSmallPrime 127 std::vector<char>::difference_type

last @0x7fffffff120 std::vector<char>::end

Welcome Edit Design Debug Projects Analyze Help

Open Documents main.cpp name_hash.cpp

name-hash

name hash.pro Headers Sources lib/StanfordCPPLib src name_hash.cpp Other files

66 67 68 69 70 71

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!
```

```
*/
```

```
int nameHash(string first, string last){
```

```
/* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than  $2^{31} - kLargePrime - 1$ .
```

```
*/
```

```
static const int kLargePrime = 16908799;
```

```
static const int kSmallPrime = 127;
```

```
int hashVal = 0;
```

```
/* Iterate across all the characters in the first name, then the last name, updating the hash at each step.
```

```
*/
```

```
for (char ch: first + last) {
```

```
/* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.
```

```
*/
```

```
ch = tolower(ch);
```

```
hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
```

```
}
```

```
return hashVal;
```

Threads: #1 name-hash

Level Function File Line

0 nameHash name_hash... 66

1 Main name_hash... 31

2 Main main.cpp 23

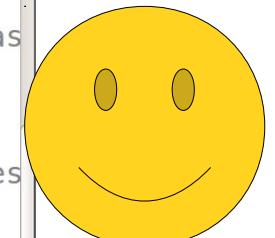
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

Once you're confident that you're on the "Step Over" button - and not the "Step Into" or "Step Out" buttons - go and click it! When you do...



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value Type

for_begin @0x7fffffff030 std::string

for_end @0x7fffffff040 std::string

for_range <not accessible> std::string

ch 97 'a' std::string

first @0x7fffffff100 std::string

hashVal 0 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::string

name hash.cpp

name-hash.pro

Headers

Sources

lib/StanfordCPPLib

src

Other files

Open Documents main.cpp name_hash.cpp

name hash.cpp

Debug

Projects

Analyze

Help

Threads: #1 name-hash

Level Function File Line

0 nameHash name_hash... 67

1 Main name_hash... 31

2 main main.cpp 23

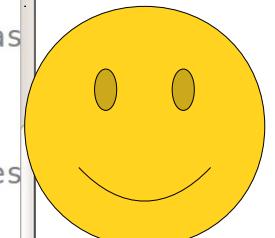
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

...your window should look something like this.



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 97 'a'

first @0x7fffffff100 std::string::iterator

hashVal 0 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::string::iterator

name hash.cpp

name hash.pro

Headers

Sources

lib/StanfordCPPLib

src

Other files

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash

Level Function File Line

0 nameHash name_hash... 67

1 Main name_hash... 31

2 Main main.cpp 23

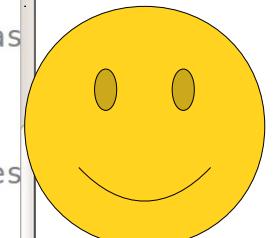
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

Okay! A few things have changed. Let's see what's going on.



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 97 'a'

first @0x7fffffff100 std::string::iterator

hashVal 0 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::string::iterator

name hash

name.hash.pro

Headers

Sources

lib/StanfordCPPLib

src

name_hash.cpp

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = **tolower(ch);**

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash

Level Function File Line

0 nameHash name_hash... 67

1 Main name_hash... 31

2 Main main.cpp 23

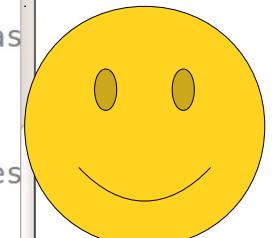
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

First, notice that our helpful Yellow Arrow friend is now pointing at line 67.



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 97 'a'

first @0x7fffffff100 std::string::iterator

hashVal 0 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::string::iterator

name hash

name.hash.pro

Headers

Sources

lib/StanfordCPPLib

src

name_hash.cpp

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for** (**char** ch: first + last) {

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = tolower(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 **return** hashVal;

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash

Level Function File Line

0 nameHash name_hash... 67

1 Main name_hash... 31

2 Main main.cpp 23

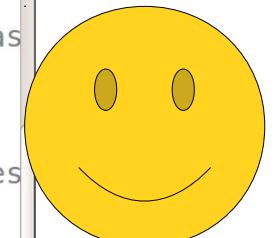
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

We're now at the line right after the one where we stopped. You just ran a single line of the program! Pretty cool!



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name	Value
_for_begin	@0x7fffffff030
_for_end	@0x7fffffff040
_for_range	<not accessible>
ch	97 'a'
first	@0x7fffffff100
hashVal	0
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff120

Welcome Edit Design Debug Projects Analyze Help

name-hash name.hash.pro Headers Sources lib/StanfordCPPLib src name_hash.cpp Other files

Open Documents main.cpp name_hash.cpp

name-hash Debug

Threads: #1 name-hash

Level	Function	File	Line
0	nameHash	name_hash...	67
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

So what did that line of code do?

```
45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - kLargePrime - 1.
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57     int hashVal = 0;
58
59     /* Iterate across all the characters in the first name, then the last
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values for
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

name hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54 static const int kLargePrime = 16908799;
55 static const int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric values
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030
_for_end @0x7fffffff040
_for_range <not accessible>
ch 97 'a'
first @0x7fffffff100
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffff120

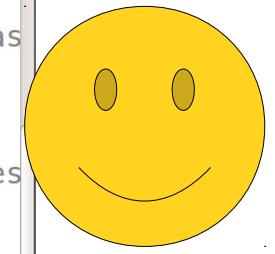
Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash

Level Function File Line

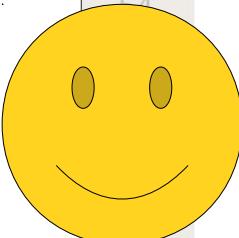
0 nameHash name_hash... 67
1 Main name_hash... 31
2 Main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages



This line converts `ch` to lower case. The `tolower` function takes in a character and returns a lower-case version of it, so this overwrites `ch` with a lower-case version of itself.

You can actually see this by looking at the values panel over on the side!



```
* F_p, where p is a large prime number, and evaluates that polynomial a
* for CS
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;
    int hashVal = 0;

    /* Iterate across all the characters in the first name, then the last
     * name, updating the hash at each step.
     */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric values
         * lower-case letters are always less than 127.
         */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

name hash
name.hash.pro
Headers
Sources
lib/StanfordCPPLib
src
Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 *
47
48
49
50
51
52
53
54

Notice that the value associated with ch has changed from 'A' to 'a' - it's now in lower-case!

STATIC
static
int kLargePrime = 16908799;
int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last name, updating the hash at each step.
*/
for (**char** ch: first + last) {
 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

Open Documents main.cpp name_hash.cpp

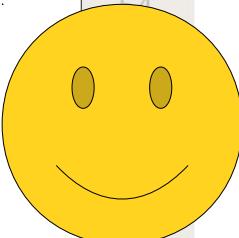
Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

1 2 3 4 5 6

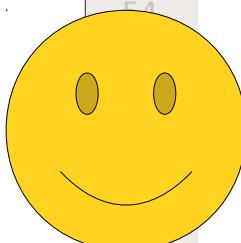
Views



For_begin @0x7fffffe030
For_end @0x7fffffe040
For_range <not accessible>
ch 97 'a'
First @0x7fffffe100
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffe120



If you'll notice, this value is in red while all the other values are in black.



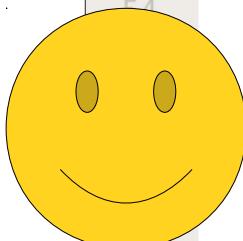
```
static int kSmallPrime = 127;
static int kLargePrime = 10900759;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```



This indicates that the value here has changed since the previous step. This is a really useful way to keep track of what's changing as you run the program.

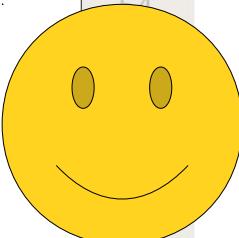


```
static
static int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

Now, let's take a look at line 67, where we are right now.



```
* F_p, where p is a large prime number, and evaluates that polynomial a
* for CS
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

static
static
int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values
 * lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

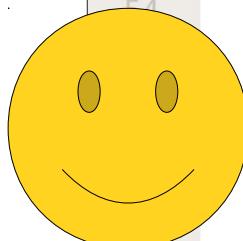
Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	67	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



Not gonna lie, this is a pretty dense line of code. It performs some weird sort of mathematical calculation on a bunch of different values.



```
static int kSmallPrime = 127;
static int kLargePrime = 10900759;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

The screenshot shows the Valgrind interface. The top bar displays "Threads: #1 name=hash" and "Stopped: 'end-stepping-range'". The main window has two tables. The left table lists threads with their file names and line numbers:

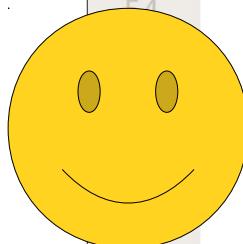
File	Line
name_hash...	67
name_hash...	31
main.cpp	23
platform.cpp	2208
name_hash...	27

The right table shows the stopped thread details:

Number	Function	File	Line	Address	Condition	Ignore	Threads (all)
1	nameHash(...	/home/keit...	66	0x4c9cc3			



Fundamentally, though, it's just computing some weird function of some values and stashing it into `hashVal`.



```
static long kLargePrime = 109008799;
static int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

Threads: #1 name-hash ◆ Stopped: "end-stepping-range".		Number	Function	File	Line	Address
n	name_hash... 67	1	nameHash(...	/home/keit...	66	0x4c9c3
	name_hash... 31					
	main.cpp 23					
ain	platform.cpp 2208					
	name hash... 27					

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

45 * F_p, where p is a large prime number, and evaluates that polynomial a
46 * for CS
47
48
49
50
51
52
53
54

Let's go run that line of code and see what happens!

STATIC
STATIC
int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last name, updating the hash at each step.
*/
for (char ch: first + last) {
 /* Convert the input character to lower case. The numeric values of lower-case letters are always less than 127.
 */
 ch = tolower(ch);
 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;

name hash.cpp

Open Documents main.cpp name hash.cpp

Threads: #1 name hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name hash...	67	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

1 2 3 4 5 6

Name Value Type
_for_begin @0x7fffffff030 std::string
_for_end @0x7fffffff040 std::string
_for_range <not accessible> std::string
ch 97 'a'
first @0x7fffffff100 std::string
hashVal 0 int
kLargePrime 16908799 int
kSmallPrime 127 int
last @0x7fffffff120 std::string

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 97 'a'

first @0x7fffffff100 std::string::iterator

hashVal 0 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::string::iterator

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for** (**char** ch: first + last) {

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = **tolower**(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 }

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

72 }

73 }

74 }

75 }

76 }

77 }

78 }

79 }

80 }

81 }

82 }

83 }

84 }

85 }

86 }

87 }

88 }

89 }

90 }

91 }

92 }

93 }

94 }

95 }

96 }

97 }

98 }

99 }

100 }

101 }

102 }

103 }

104 }

105 }

106 }

107 }

108 }

109 }

110 }

111 }

112 }

113 }

114 }

115 }

116 }

117 }

118 }

119 }

120 }

121 }

122 }

123 }

124 }

125 }

126 }

127 }

128 }

129 }

130 }

131 }

132 }

133 }

134 }

135 }

136 }

137 }

138 }

139 }

140 }

141 }

142 }

143 }

144 }

145 }

146 }

147 }

148 }

149 }

150 }

151 }

152 }

153 }

154 }

155 }

156 }

157 }

158 }

159 }

160 }

161 }

162 }

163 }

164 }

165 }

166 }

167 }

168 }

169 }

170 }

171 }

172 }

173 }

174 }

175 }

176 }

177 }

178 }

179 }

180 }

181 }

182 }

183 }

184 }

185 }

186 }

187 }

188 }

189 }

190 }

191 }

192 }

193 }

194 }

195 }

196 }

197 }

198 }

199 }

200 }

201 }

202 }

203 }

204 }

205 }

206 }

207 }

208 }

209 }

210 }

211 }

212 }

213 }

214 }

215 }

216 }

217 }

218 }

219 }

220 }

221 }

222 }

223 }

224 }

225 }

226 }

227 }

228 }

229 }

230 }

231 }

232 }

233 }

234 }

235 }

236 }

237 }

238 }

239 }

240 }

241 }

242 }

243 }

244 }

245 }

246 }

247 }

248 }

249 }

250 }

251 }

252 }

253 }

254 }

255 }

256 }

257 }

258 }

259 }

260 }

261 }

262 }

263 }

264 }

265 }

266 }

267 }

268 }

269 }

270 }

271 }

272 }

273 }

274 }

275 }

276 }

277 }

278 }

279 }

280 }

281 }

282 }

283 }

284 }

285 }

286 }

287 }

288 }

289 }

290 }

291 }

292 }

293 }

294 }

295 }

296 }

297 }

298 }

299 }

300 }

301 }

302 }

303 }

304 }

305 }

306 }

307 }

308 }

309 }

310 }

311 }

312 }

313 }

314 }

315 }

316 }

317 }

318 }

319 }

320 }

321 }

322 }

323 }

324 }

325 }

326 }

327 }

328 }

329 }

330 }

331 }

332 }

333 }

334 }

335 }

336 }

337 }

338 }

339 }

340 }

341 }

342 }

343 }

344 }

345 }

346 }

347 }

348 }

349 }

350 }

351 }

352 }

353 }

354 }

355 }

356 }

357 }

358 }

359 }

360 }

361 }

362 }

363 }

364 }

365 }

366 }

367 }

368 }

369 }

370 }

371 }

372 }

373 }

374 }

375 }

376 }

377 }

378 }

379 }

380 }

381 }

382 }

383 }

384 }

385 }

386 }

387 }

388 }

389 }

390 }

391 }

392 }

393 }

394 }

395 }

396 }

397 }

398 }

399 }

400 }

401 }

402 }

403 }

404 }

405 }

406 }

407 }

408 }

409 }

410 }

411 }

412 }

413 }

414 }

415 }

416 }

417 }

418 }

419 }

420 }

421 }

422 }

423 }

424 }

425 }

426 }

427 }

428 }

429 }

430 }

431 }

432 }

433 }

434 }

435 }

436 }

437 }

438 }

439 }

440 }

441 }

442 }

443 }

444 }

445 }

446 }

447 }

448 }

449 }

450 }

451 }

452 }

453 }

454 }

455 }

456 }

457 }

458 }

459 }

460 }

461 }

462 }

463 }

464 }

465 }

466 }

467 }

468 }

469 }

470 }

471 }

472 }

473 }

474 }

475 }

476 }

477 }

478 }

479 }

480 }

481 }

482 }

483 }

484 }

485 }

486 }

487 }

488 }

489 }

490 }

491 }

492 }

493 }

494 }

495 }

496 }

497 }

498 }

499 }

500 }

501 }

502 }

503 }

504 }

505 }

506 }

507 }

508 }

509 }

510 }

511 }

512 }

513 }

514 }

515 }

516 }

517 }

518 }

519 }

520 }

521 }

522 }

523 }

524 }

525 }

526 }

527 }

528 }

529 }

530 }

531 }

532 }

533 }

534 }

535 }

536 }

537 }

538 }

539 }

540 }

541 }

542 }

543 }

544 }

545 }

546 }

547 }

548 }

549 }

550 }

551 }

552 }

553 }

554 }

555 }

556 }

557 }

558 }

559 }

560 }

561 }

562 }

563 }

564 }

565 }

566 }

567 }

568 }

569 }

570 }

571 }

572 }

573 }

574 }

575 }

576 }

577 }

578 }

579 }

580 }

581 }

582 }

583 }

584 }

585 }

586 }

587 }

588 }

589 }

590 }

591 }

592 }

593 }

594 }

595 }

596 }

597 }

598 }

599 }

600 }

601 }

602 }

603 }

604 }

605 }

606 }

607 }

608 }

609 }

610 }

611 }

612 }

613 }

614 }

615 }

616 }

617 }

618 }

619 }

620 }

621 }

622 }

623 }

624 }

625 }

626 }

627 }

628 }

629 }

630 }

631 }

632 }

633 }

634 }

635 }

636 }

637 }

638 }

639 }

640 }

641 }

642 }

643 }

644 }

645 }

646 }

647 }

648 }

649 }

650 }

651 }

652 }

653 }

654 }

655 }

656 }

657 }

658 }

659 }

660 }

661 }

662 }

663 }

664 }

665 }

666 }

667 }

668 }

669 }

670 }

671 }

672 }

673 }

674 }

675 }

676 }

677 }

678 }

679 }

680 }

681 }

682 }

683 }

684 }

685 }

686 }

687 }

688 }

689 }

690 }

691 }

692 }

693 }

694 }

695 }

696 }

697 }

698 }

699 }

700 }

701 }

702 }

703 }

704 }

705 }

706 }

707 }

708 }

709 }

710 }

711 }

712 }

713 }

714 }

715 }

716 }

717 }

718 }

719 }

720 }

721 }

722 }

723 }

724 }

725 }

726 }

727 }

728 }

729 }

730 }

731 }

732 }

733 }

734 }

735 }

736 }

737 }

738 }

739 }

740 }

741 }

742 }

743 }

744 }

745 }

746 }

747 }

748 }

749 }

750 }

751 }

752 }

753 }

754 }

755 }

756 }

757 }

758 }

759 }

760 }

761 }

762 }

763 }

764 }

765 }

766 }

767 }

768 }

769 }

770 }

771 }

772 }

773 }

774 }

775 }

776 }

777 }

778 }

779 }

780 }

781 }

782 }

783 }

784 }

785 }

786 }

787 }

788 }

789 }

790 }

791 }

792 }

793 }

794 }

795 }

796 }

797 }

798 }

799 }

800 }

801 }

802 }

803 }

804 }

805 }

806 }

807 }

808 }

809 }

810 }

811 }

812 }

813 }

814 }

815 }

816 }

817 }

818 }

819 }

820 }

821 }

822 }

823 }

824 }

825 }

826 }

827 }

828 }

829 }

830 }

831 }

832 }

833 }

834 }

835 }

836 }

837 }

838 }

839 }

840 }

841 }

842 }

843 }

844 }

845 }

846 }

847 }

848 }

849 }

850 }

851 }

852 }

853 }

854 }

855 }

856 }

857 }

858 }

859 }

860 }

861 }

862 }

863 }

864 }

865 }

866 }

867 }

868 }

869 }

870 }

871 }

872 }

873 }

874 }

875 }

876 }

877 }

878 }

879 }

880 }

881 }

882 }

883 }

884 }

885 }

886 }

887 }

888 }

889 }

890 }

891 }

892 }

893 }

894 }

895 }

896 }

897 }

898 }

899 }

900 }

901 }

902 }

903 }

904 }

905 }

906 }

907 }

908 }

909 }

910 }

911 }

912 }

913 }

914 }

915 }

916 }

917 }

918 }

919 }

920 }

921 }

922 }

923 }

924 }

925 }

926 }

927 }

928 }

929 }

930 }

931 }

932 }

933 }

934 }

935 }

936 }

937 }

938 }

939 }

940 }

941 }

942 }

943 }

944 }

945 }

946 }

947 }

948 }

949 }

950 }

951 }

952 }

953 }

954 }

955 }

956 }

957 }

958 }

959 }

960 }

961 }

962 }

963 }

964 }

965 }

966 }

967 }

968 }

969 }

970 }

971 }

972 }

973 }

974 }

975 }

976 }

977 }

978 }

979 }

980 }

981 }

982 }

983 }

984 }

985 }

986 }

987 }

988 }

989 }

990 }

991 }

992 }

993 }

994 }

995 }

996 }

997 }

998 }

999 }

1000 }

1001 }

1002 }

1003 }

1004 }

1005 }

1006 }

1007 }

1008 }

1009 }

1010 }

1011 }

1012 }

1013 }

1014 }

1015 }

1016 }

1017 }

1018 }

1019 }

1020 }

1021 }

1022 }

1023 }

1024 }

1025 }

1026 }

1027 }

1028 }

1029 }

1030 }

1031 }

1032 }

1033 }

1034 }

1035 }

1036 }

1037 }

1038 }

1039 }

1040 }

1041 }

1042 }

1043 }

1044 }

1045 }

1046 }

1047 }

1048 }

1049 }

1050 }

1051 }

1052 }

1053 }

1054 }

1055 }

1056 }

1057 }

1058 }

1059 }

1060 }

1061 }

1062 }

1063 }

1064 }

1065 }

1066 }

1067 }

1068 }

1069 }

1070 }

1071 }

1072 }

1073 }

1074 }

1075 }

1076 }

1077 }

1078 }

1079 }

1080 }

1081 }

1082 }

1083 }

1084 }

1085 }

1086 }

1087 }

1088 }

1089 }

1090 }

1091 }

1092 }

1093 }

1094 }

1095 }

1096 }

1097 }

1098 }

1099 }

1100 }

1101 }

1102 }

1103 }

1104 }

1105 }

1106 }

1107 }

1108 }

1109 }

1110 }

1111 }

1112 }

1113 }

1114 }

1115 }

1116 }

1117 }

1118 }

1119 }

1120 }

1121 }

1122 }

1123 }

1124 }

1125 }

1126 }

1127 }

1128 }

1129 }

1130 }

1131 }

1132 }

1133 }

1134 }

1135 }

1136 }

1137 }

1138 }

1139 }

1140 }

1141 }

1142 }

1143 }

1144 }

1145 }

1146 }

1147 }

1148 }

1149 }

1150 }

1151 }

1152 }

1153 }

1154 }

1155 }

1156 }

1157 }

1158 }

1159 }

1160 }

1161 }

1162 }

1163 }

1164 }

1165 }

1166 }

1167 }

1168 }

1169 }

1170 }

1171 }

1172 }

1173 }

1174 }

1175 }

1176 }

1177 }

1178 }

1179 }

1180 }

1181 }

1182 }

1183 }

1184 }

1185 }

1186 }

1187 }

1188 }

1189 }

1190 }

1191 }

1192 }

1193 }

1194 }

1195 }

1196 }

1197 }

1198 }

1199 }

1200 }

1201 }

1202 }

1203 }

1204 }

1205 }

1206 }

1207 }

1208 }

1209 }

1210 }

1211 }

1212 }

1213 }

1214 }

1215 }

1216 }

1217 }

1218 }

1219 }

1220 }

1221 }

1222 }

1223 }

1224 }

1225 }

1226 }

1227 }

1228 }

1229 }

1230 }

1231 }

1232 }

1233 }

1234 }

1235 }

1236 }

1237 }

1238 }

1239 }

1240 }

1241 }

1242 }

1243 }

1244 }

1245 }

1246 }

1247 }

1248 }

1249 }

1250 }

1251 }

1252 }

1253 }

1254 }

1255 }

1256 }

1257 }

1258 }

1259 }

1260 }

1261 }

1262 }

1263 }

1264 }

1265 }

1266 }

1267 }

1268 }

1269 }

1270 }

1271 }

1272 }

1273 }

1274 }

1275 }

1276 }

1277 }

1278 }

1279 }

1280 }

1281 }

1282 }

1283 }

1284 }

1285 }

1286 }

1287 }

1288 }

1289 }

1290 }

1291 }

1292 }

1293 }

1294 }

1295 }

1296 }

1297 }

1298 }

1299 }

1300 }

1301 }

1302 }

1303 }

1304 }

1305 }

1306 }

1307 }

1308 }

1309 }

1310 }

1311 }

1312 }

1313 }

1314 }

1315 }

1316 }

1317 }

1318 }

1319 }

1320 }

1321 }

1322 }

1323 }

1324 }

1325 }

1326 }

1327 }

1328 }

1329 }

1330 }

1331 }

1332 }

1333 }

1334 }

1335 }

1336 }

1337 }

1338 }

1339 }

1340 }

1341 }

1342 }

1343 }

1344 }

1345 }

1346 }

1347 }

1348 }

1349 }

1350 }

1351 }

1352 }

1353 }

1354 }

1355 }

1356 }

1357 }

1358 }

1359 }

1360 }

1361 }

1362 }

1363 }

1364 }

1365 }

1366 }</p

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

Name Value Type

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 97 'a' std::string::iterator

first 97 std::string::iterator

hashVal 16908799 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff0120 std::string::iterator

Welcome Edit Design Debug Projects Analyze Help

name hash name.hash.pro Headers Sources lib/StanfordCPPLib src name_hash.cpp Other files

Open Documents main.cpp name_hash.cpp

name hash

Debug

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = tolower(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 **return** hashVal;

64 }

65

66

67

68

69

70

71 }

Threads: #1 nar

Level Function File L

0 nameHash name_hash... 62

1 Main name_hash... 31

2 Main main.cpp 23

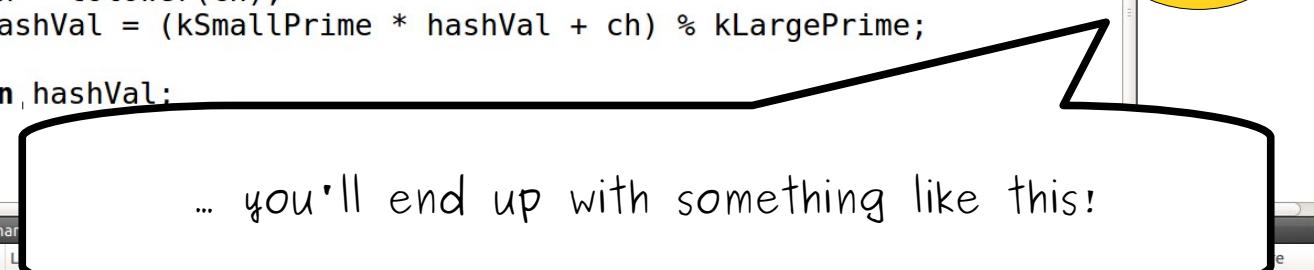
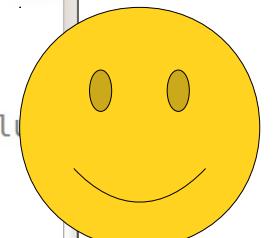
3 startupMain platform.cpp 2208

4 main name hash... 27

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

you'll end up with something like this!



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

Name Value Type

_for_begin @0x7fffffff030 std::string::iterator

_for_end @0x7fffffff040 std::string::iterator

_for_range <not accessible> std::string::iterator

ch 97 'a' std::string::iterator

first 97 std::string::iterator

hashVal 16908799 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff0120 std::string::iterator

name hash.cpp

name hash.pro

Headers

Sources

lib/StanfordCPPLib

src

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashVal = 0;

54

55 /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.

56 */

57 **for (char ch: first + last) {**

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = **tolower(ch);**

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 **return** hashVal;

64 }

65 }

66

67

68

69

70

71 }

Open Documents main.cpp name_hash.cpp

Threads: #1 nar

Level Function File L

0 nameHash name_hash... 62

1 Main name_hash... 31

2 Main main.cpp 23

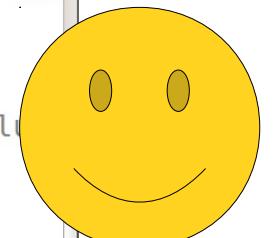
3 startupMain platform.cpp 2208

4 main name hash... 27

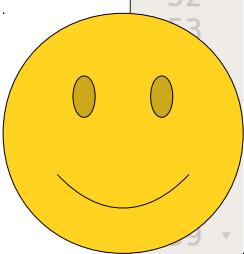
Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

Let's see what's changed.



First, notice that the value stored in `hashVal` changed to 97. We know that it changed because the value is in red, and we know that nothing else changed because nothing else is in red!



```
static const int kLargePrime = 16908799;
static const int kSmallPrime = 127;

int hashVal = 0;

/* Iterate across all the characters in the first name, then the last
 * name, updating the hash at each step.
 */
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric values
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

The screenshot shows a debugger interface with a yellow smiley face icon overlaid on the code editor area. A callout bubble contains handwritten text about the variable `hashVal`. The right side of the screen displays the Qt Creator interface, including the Projects, Sources, and Open Documents panes, the code editor with the highlighted function, the Registers and Stack frames, and the Threads and Callstack tabs. The Registers tab shows memory dump information, and the Callstack tab shows the current stack trace.

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

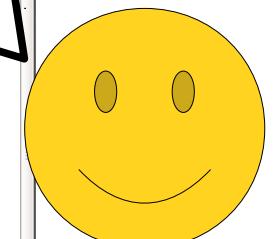
Line: 62, Col: 5

Name	Value
_for_begin	@0x7fffffff030
_for_end	@0x7fffffff040
_for_range	<not accessible>
ch	97 'a'
first	
hashVal	97
kLargePrime	16908799
kSmallPrime	127
last	@0x7fffffff0120

```

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers, a large prime and a
51     * prime. These numbers were chosen because their product is less than
52     * 2^31 - 1.
53     */
54     static const int kSmallPrime = 127;
55     static const int kLargePrime = 16908799;
56
57     int hashVal = 97;
58
59     /* Iterate over each character in the string, updating the hash value.
60     * name, up to last.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case. The numeric values for
64         * lower-case letters are always less than 127.
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

Second, notice that we're back up at the top of the for loop, since that's where the yellow arrow is pointing. We ended up back here because this is the next line that gets executed.



Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash... 62		1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash... 31									
2	Main	main.cpp 23									
3	startupMain	platform.cpp 2208									
4	main	name hash... 27									

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

Name Value

_for_begin @0x7fffffff030 std::for_each::iterator

_for_end @0x7fffffff040 std::for_each::iterator

_for_range <not accessible> std::for_each::range

ch 97 'a'

first @0x7fffffff100 std::vector<char>::iterator

hashVal 97 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff120 std::vector<char>::iterator

name hash.cpp

name hash.pro

Headers

Sources

lib/StanfordCPPLib

src

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47

48

49 **int** nameHash(**string** first, **string** last){

50 /* This hashing scheme needs two prime numbers, a large prime and a

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 **static const int** kLargePrime = 16908799;

55 **static const int** kSmallPrime = 127;

56

57

58

59

60

61 **int** hashVal;

62 /* Iterates over every character in the input strings. The numeric values of lower-case letters are always less than 127.

63 * name,

64 */

65

66 **for** (**char** ch: first + last) {

67 /* Convert the input character to lower case. The numeric values of lower-case letters are always less than 127.

68 */

69 ch = tolower(ch);

70 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

71 }

72 **return** hashVal;

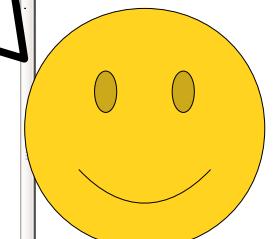
73 }

We just single-stepped through a single iteration of that loop! Pretty cool!

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash... 62		1	nameHash(...	/home/keit... 66		0x4c9cc3			(all)
1	Main	name_hash... 31									
2	Main	main.cpp 23									
3	startupMain	platform.cpp 2208									
4	main	name hash... 27									

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

Name Value

_for_begin @0x7fffffff030 std::for_each::begin

_for_end @0x7fffffff040 std::for_each::end

_for_range <not accessible> std::for_each::range

ch 97 'a'

first 97

hashVal 16908799

kLargePrime 16908799

kSmallPrime 127

last @0x7fffffff0120 std::for_each::last

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashV

54 /* Iterat

55 * name,

56 */

57 **for** (**char** ch: first + last) {

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = tolower(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 **return** hashVal;

64 }

65

66

67

68

69

70

71 }

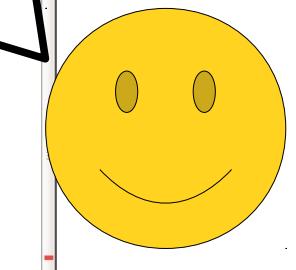
Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash...	62	1	nameHash(...	/home/keit...	66	0x4c9cc3	(all)		
1	Main	name_hash...	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

Let's go do it again!



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

Name Value

_for_begin @0x7fffffff030
_for_end @0x7fffffff040
_for_range <not accessible>
ch 97 'a'
first
hashVal
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffff000

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54 static const int kLargePrime = 16908799;
55 static const int kSmallPrime = 127;
56
int hashV
/* Iterat
* name,
*/
57
58
59
60
61
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric values for
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70
71 }

Again, move your mouse over the Step Over button (and make sure it says "Step Over" and not something else!), then click it.

Threads: #1 name-hash Stopped: "end-stepping-range".

Level Function File Line

0 nameHash name_hash... 62
1 Main name_hash... 31
2 Main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

name hash
name.hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kLargePrime = 16908799;

51 **static const int** kSmallPrime = 127;

52

53 **int** hashV

54 /* Iterat

55 * name,

56 */

57 **for** (**char** ch: first + last) {

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = tolower(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 **return** hashVal;

64 }

65

66

67

68

69

70

71 }

Line: 66, Col: 9

Name Value

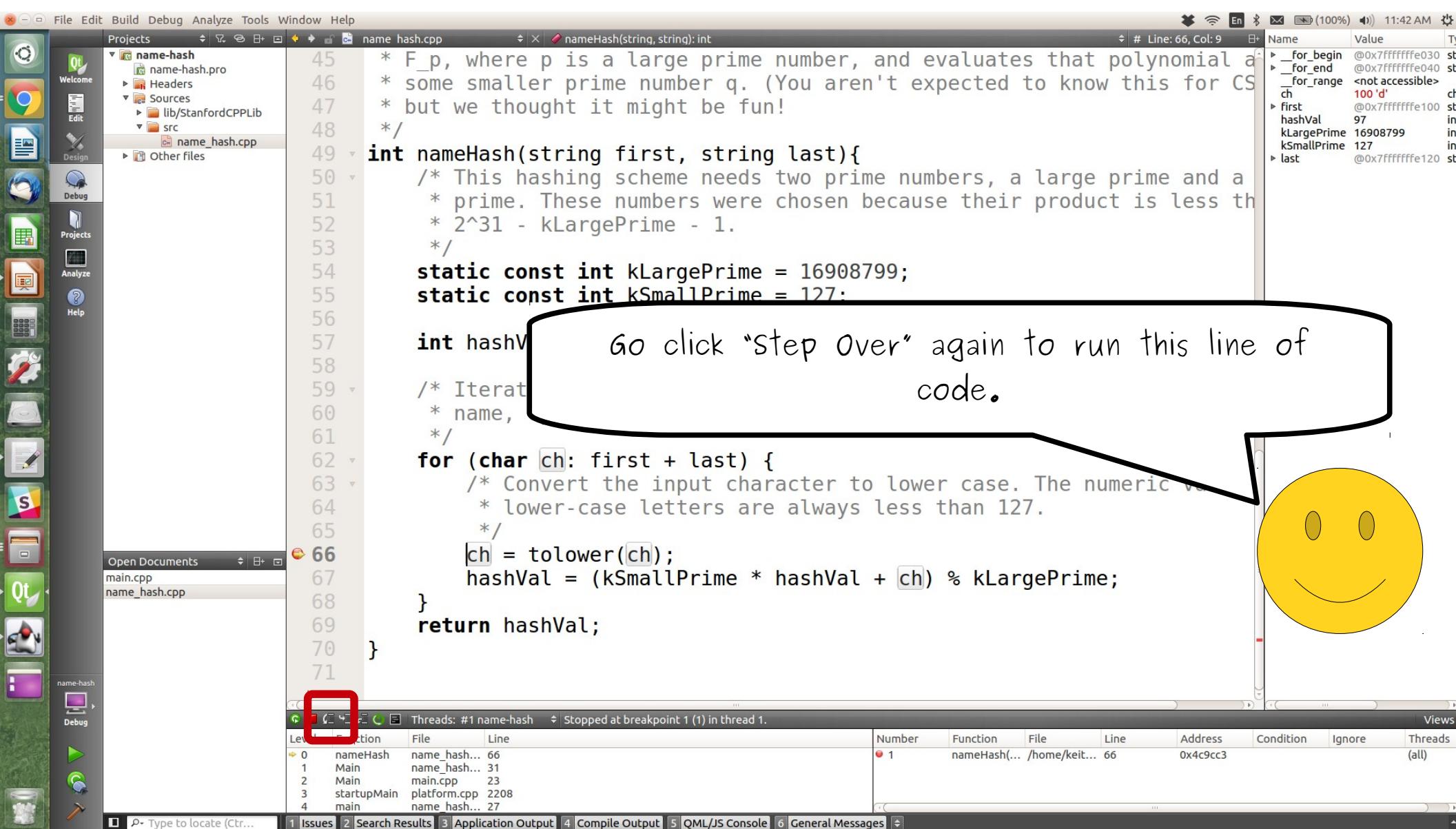
_for_begin @0x7fffffe030
_for_end @0x7fffffe040
_for_range <not accessible>
ch 100 'd'
First @0x7fffffe100
hashVal 97
kLargePrime 16908799
kSmallPrime 127
last @0x7fffffe120

Now we're here! Notice that ch now has the value 'd', which is the second letter of the name Ada.

Threads: #1 name-hash Stopped at breakpoint 1 (1) in thread 1.

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash... 66		1	nameHash(...	/home/keit...	66	0x4c9cc3			(all)
1	Main	name_hash... 31									
2	Main	main.cpp 23									
3	startupMain	platform.cpp 2208									
4	main	name hash... 27									

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030 std::for_each::begin

_for_end @0x7fffffff040 std::for_each::end

_for_range <not accessible> std::for_each::range

ch 100 'd'

first 97 std::vector<char>::iterator

hashVal 16908799 int

kLargePrime 16908799 int

kSmallPrime 127 int

last @0x7fffffff0120 std::vector<char>::back

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47

48

49 **int** nameHash(**string** first, **string** last){

50 /* This hashing scheme needs two prime numbers, a large prime and a

51 * prime. These numbers were chosen because their product is less than

52 * $2^{31} - kLargePrime - 1$.

53 */

54 **static const int** kSmallPrime = 16908799;

55 **static const int** kLargePrime = 16908799;

56

57 **int** hashVal = 97;

58

59 /* Iterate over every character in the input strings.

60 * name, last.

61 */

62 **for** (**char** ch: first + last) {

63 /* Convert the input character to lower case. The numeric values for

64 * lower-case letters are always less than 127.

65 */

66 ch = **tolower**(ch);

67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

68 }

69

70 **return** hashVal;

71 }

You should be here now. Notice that none of the values changed. That makes sense, since all we did was convert a lower-case 'd' to a lower-case 'd'.

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	nameHash	name_hash.cpp	67	1	nameHash(...	/home/keith...	66	0x4c9cc3			(all)
1	Main	name_hash.cpp	31								
2	Main	main.cpp	23								
3	startupMain	platform.cpp	2208								
4	main	name hash...	27								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 67, Col: 9

Name Value

_for_begin @0x7fffffff030 std::for_each::begin

_for_end @0x7fffffff040 std::for_each::end

_for_range <not accessible> std::for_each::range

ch 100 'd'

first 97 std::string::front

kLargePrime 16908799 std::int

kSmallPrime 127 std::int

last @0x7fffffff0120 std::string::back

45 * F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!

46 */

47 **int** nameHash(**string** first, **string** last){

48 /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than $2^{31} - kLargePrime - 1$.

49 */

50 **static const int** kSmallPrime = 127;

51 **static const int** kLargePrime = 16908799;

52

53 **int** hashVal = 1;

54 /* Iterates over each character in the string. If the character is uppercase, converts it to lowercase.

55 * name, and adds it to the hash value.

56 */

57 **for** (**char** ch: first + last) {

58 /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.

59 */

60 ch = tolower(ch);

61 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;

62 }

63 **return** hashVal;

64 }

65 }

66 }

67 }

68 }

69 }

70 }

71 }

Threads: #1 name-hash Stopped: "end-stepping-range".

Level Function File Line

0 nameHash name_hash... 67

1 Main name_hash... 31

2 Main main.cpp 23

3 startupMain platform.cpp 2208

4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

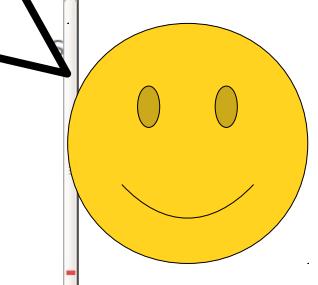
1 nameHash(... /home/keit... 66 0x4c9cc3 (all)

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages

Now, click "Step Over" one more time.

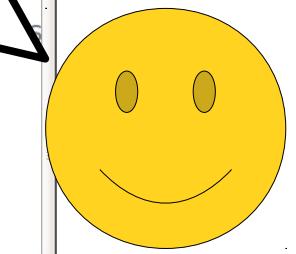
You'll now be at this point in the program. We've covered up the value of `hashVal` in this image, because at this point you should be able to see what `hashVal` is by reading the value in the side pane. This is the special value we want you to tell us when submitting the assignment!



```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!
*/
int
/*
for (char ch: first + last) {
    /* Convert the input character to lower case. The numeric
     * lower-case letters are always less than 127.
     */
    ch = tolower(ch);
    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
}
return hashVal;
```

The screenshot shows a Qt Creator IDE interface. The left sidebar contains icons for Welcome, Edit, Design, Projects, Analyze, and Help. The main window has a title bar with "File Edit Build Debug Analyze Tools Window Help". The "Projects" tab is selected, showing a project named "name-hash" with files like "name.hash.pro", "Headers", "Sources", "lib/StanfordCPPLib", and "src/name_hash.cpp". Below the projects is an "Open Documents" section with "main.cpp" and "name_hash.cpp". The "name_hash.cpp" file is open in the center, showing C++ code for a polynomial hash function. A callout bubble points from the text "This is the special value we want you to tell us when submitting the assignment!" to the variable "hashVal" in the code. To the right of the code, there is a "Variables" view showing memory addresses and values for variables like "first", "hashVal", "kLargePrime", and "kSmallPrime". A red box highlights the question mark icon next to the value of "hashVal". At the bottom, there is a "Threads" view showing the execution stack and a "Views" tab.

To finish up this section on the debugger, we'd like to show you two last little techniques that you might find useful when debugging programs.



```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!
*/
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than 2^31 - kLargePrime - 1.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

The screenshot shows a debugger interface with the following details:

- Projects:** A tree view showing a project named "name-hash" with files like "name.hash.pro", "Headers", "Sources", "lib/StanfordCPPLib", and "src/name_hash.cpp".
- Code Editor:** The "name_hash.cpp" file is open, showing the implementation of the `nameHash` function.
- Registers:** A sidebar shows registers with values like `_for_begin`, `_For_end`, `_for_range`, `ch`, `first`, `hashVal`, `kLargePrime`, `kSmallPrime`, and `last`. The value for `last` is highlighted with a red box and a question mark.
- Threads:** A table titled "Threads: #1 name-hash" shows the call stack:

Level	Function	File	Line
0	nameHash	name_hash...	62
1	Main	name_hash...	31
2	Main	main.cpp	23
3	startupMain	platform.cpp	2208
4	main	name hash...	27
- Bottom Bar:** Includes tabs for "Issues", "Search Results", "Application Output", "Compile Output", "QML/JS Console", and "General Messages".

To start this off, click on the the breakpoint that we set earlier in the program. If you do.

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun!)* /int nameHash(string first, string last){    /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than 2^31 - kLargePrime - 1.* /for (char ch: first + last) {    /* Convert the input character to lower case. The numeric lower-case letters are always less than 127.    */    ch = tolower(ch);    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;}
```

The screenshot shows the Qt Creator IDE interface. On the left is the project tree with 'name-hash' selected. The main window displays the 'name_hash.cpp' file. A callout bubble originates from a yellow smiley face icon on the right side of the interface, pointing towards the toolbar. The toolbar contains various icons, including one for setting breakpoints, which is highlighted with a blue circle. The status bar at the bottom shows the current file is 'name hash.cpp' and the line number is 62. The bottom right corner shows the Qt logo.

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

Name Value

for_begin @0x7fffffff030
for_end @0x7fffffff040
for_range <not accessible>
ch 100'd'
first ?
hashVal ffffffe100
kLargePrime 99
kSmallPrime 127
last @0x7fffffff120

name hash.cpp

name hash.pro

Headers

Sources

lib/StanfordCPPLib

src

Other files

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54
55
56
57
58
59
60
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash Stopped: "end-stepping-range".

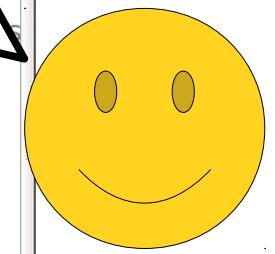
Level Function File Line

0 nameHash name_hash... 62
1 Main name_hash... 31
2 Main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27

Number Function File Line Address Condition Ignore Threads

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

it should clear the breakpoint. Now, if we were to run this program again in debug mode, it would not stop at this point, since nothing's telling it to!



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54 static const int kLargePrime = 16908799;
55 static const int kSmallPrime = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case. The numeric values for
64 * lower-case letters are always less than 127.
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }

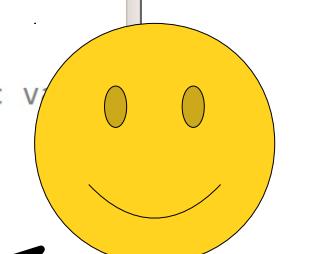
Threads: #1 name-hash

Level	Function	File	Line
0	nameHash	name_hash... 62	
1	Main	name_hash... 31	
2	Main	main.cpp 23	
3	startupMain	platform.cpp 2208	
4	main	name hash... 27	

Type to locate (Ctrl...) Issues Search Results Application O

Name Value

_for_begin @0x7fffffff030
_for_end @0x7fffffff040
_for_range <not accessible>
ch 100 'd'
first hashVal ?
kLargePrime ffffffe100
kSmallPrime 127
last @0x7fffffff0120



Now, take a look back at these buttons.

The screenshot shows a Qt-based IDE interface. On the left is a vertical toolbar with various icons for file operations, projects, and analysis. The main window has a "Projects" tab at the top, followed by a tree view of the "name-hash" project structure, including "name-hash.pro", "Headers", "Sources" (containing "lib/StanfordCPPLib" and "src"), and "Other files". Below this is an "Open Documents" section with "main.cpp" and "name_hash.cpp". The central area is a code editor with the following C++ code:

```
* F_p, where p is a large prime number, and evaluates that polynomial at
* some smaller prime number q. (You aren't expected to know this for CS
* but we thought it might be fun!
*/
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a
     * prime. These numbers were chosen because their product is less than
     * 2^31 - kLargePrime - 1.
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

    /* Iterate across all the characters in the first name, then the last
     * name, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric values for
         * lower-case letters are always less than 127.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

The code editor shows line numbers 45 through 71. A red box highlights the "Threads" button in the toolbar at the bottom left. The variable viewer on the right shows several variables with their memory addresses and values. A red box highlights the value of "last" which is "?". A large yellow smiley face is positioned on the right side of the interface. A callout bubble from the smiley face points to a tooltip containing the text: "Hover your mouse over the one that's fifth from the left. When you hover over it, it should say 'Step Out.'".

The screenshot shows a Qt IDE interface with the following components:

- Projects View:** Shows the project structure for "name-hash".
- Code Editor:** Displays the "nameHash.cpp" file content. The code implements a polynomial hashing scheme using two prime numbers, kLargePrime (16908799) and kSmallPrime (127). It iterates over the characters of the input names, converting them to lowercase using `tolower`, and updates a hash value.
- Variables View:** Shows the current state of variables. The variable "last" has a value of 99, which is highlighted with a red box and a question mark icon.
- Debugger Toolbar:** Includes buttons for running, stopping, and stepping through code. The "Run" button is highlighted with a red box.
- Threads View:** Shows the current threads, with "name-hash" as the active thread.
- Callout Bubble:** A hand-drawn style callout points from a yellow smiley face on the right towards the "Run" button in the toolbar.

Code in nameHash.cpp:

```
* F_p, where p is a large prime number, and evaluates that polynomial at some smaller prime number q. (You aren't expected to know this for CS but we thought it might be fun! */

int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers, a large prime and a prime. These numbers were chosen because their product is less than  $2^{31} - kLargePrime - 1$ .
     */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

    /* Iterate across all the characters in the first name, then the last name, updating the hash at each step.
     */
    for (char ch: first + last) {
        /* Convert the input character to lower case. The numeric values for lower-case letters are always less than 127.
         */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int

Line: 62, Col: 5

45 * F_p, where p is a large prime number, and evaluates that polynomial at
46 some smaller prime number q. (You aren't expected to know this for CS
47 but we thought it might be fun!
48 */

49 **int** nameHash(**string** first, **string** last){
50 /* This hashing scheme needs two prime numbers, a large prime and a
51 * prime. These numbers were chosen because their product is less than
52 * $2^{31} - kLargePrime - 1$.
53 */
54 **static const int** kLargePrime = 16908799;
55 **static const int** kSmallPrime = 127;

56
57 **int** hashVal = 0;

58
59 /* Iterate across all the characters in the first name, then the last
60 * name, updating the hash at each step.
61 */
62 **for** (**char** ch: first + last) {
63 /* Convert the input character to lower case. The numeric values for
64 * lower-case letters are always less than 127.
65 */
66 ch = **tolower**(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 **return** hashVal;
70
71 }

Threads: #1 name-hash

Level File Line

0 nameHash name_hash... 62
1 Main name_hash... 31
2 Main main.cpp 23
3 startupMain platform.cpp 2208
4 main name hash... 27

Type to locate (Ctrl+F)

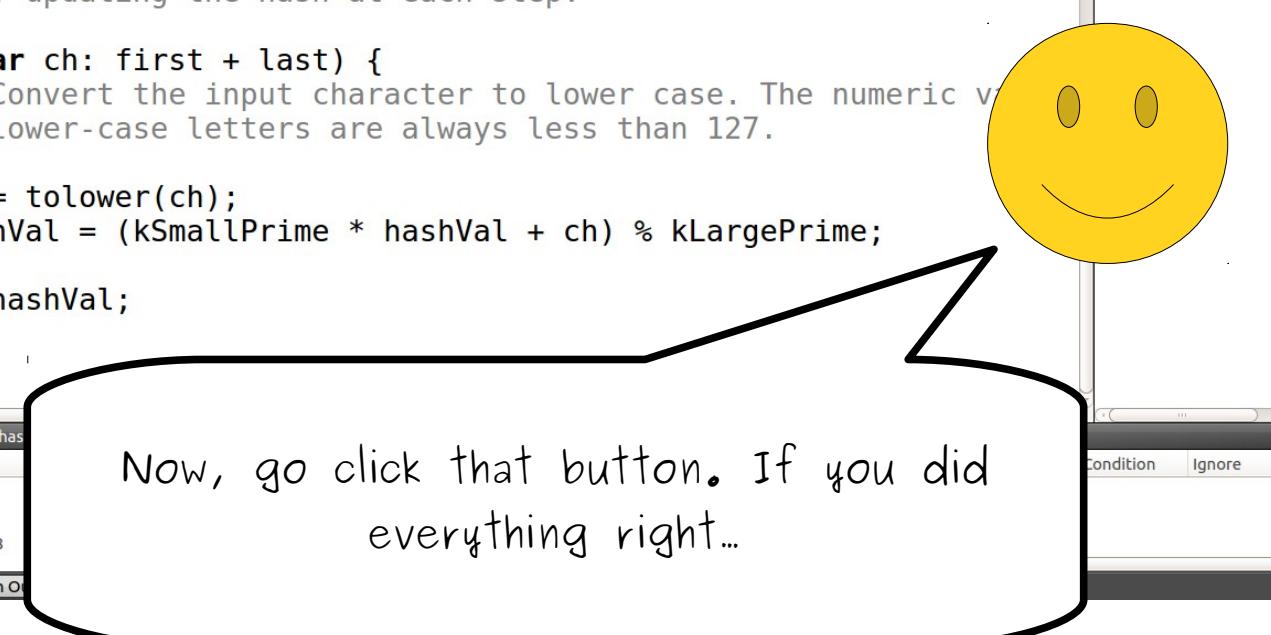
Issues Search Results Application Output

Name Value

_for_begin @0x7fffffe030
_for_end @0x7fffffe040
_for_range <not accessible>
ch 100 'd'
first hashVal ffffffe100
kLargePrime 99
kSmallPrime 127
last @0x7fffffe120

?

Now, go click that button. If you did everything right...



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int # Line: 31, Col: 5 Name Value

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

```
#include "console.h"
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the act
 * to talk more
 * the meaning
 * of the input
 *
 * For those
 * treats each
 * It then uses
 * F n where n is
```

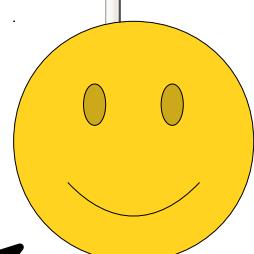
... you should end up with something that looks like this!

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

Views

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages



File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int # Line: 31, Col: 5

name-hash
name.hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

```
#include "console.h"
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the act
 * to talk mo
 * the meanti
 * of the inp
 *
 * For those
 * treats eac
 * It then us
 * F n where n
```

Open Documents main.cpp name_hash.cpp

Threads: #1 name-hash Stopped: "function-finished".

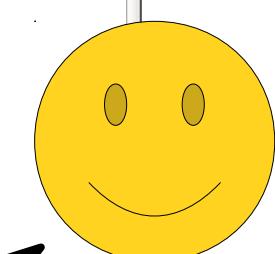
Level	Function	File	Line
0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

Name Value

first @0x7fffffff0ao
last @0x7fffffff0eoC0
hashValue -590633613

returned value 1967457

Let's take a minute to get our bearings.
Where exactly are we?



The screenshot shows a Qt IDE interface with the following details:

- File Menu:** File, Edit, Build, Debug, Analyze, Tools, Window, Help.
- Projects View:** Shows a project named "name-hash" with files "name-hash.pro", "Headers", "Sources" (containing "lib/StanfordCPPLib" and "src"), and "name_hash.cpp".
- Code Editor:** Displays the "name_hash.cpp" file with the following code:

```
#include "console.h"
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the act
 * to talk mo
 * the meati
 * of the inp
 *
 * For those
 * treats eac
 * It then us
 * F n where n
```
- Variables View:** Shows variables "first" (0x7fffffff0a0), "last" (0x7fffffff0c0), and "hashValue" (-590633613).
- Output View:** Shows the returned value 1967457.
- Threads View:** Shows threads: #1 name-hash Stopped: "function-finished".
- Status Bar:** Type to locate (Ctrl...), Issues, Search Results, Application Output, Compile Output, QML/JS Console, General Messages.

A large yellow arrow points from a yellow smiley face on the right towards a callout bubble containing handwritten text:

Well, the yellow arrow indicates that we're back in `main` again. Cool!

File Edit Build Debug Analyze Tools Window Help

Projects name hash.cpp nameHash(string, string): int # Line: 31, Col: 5

name-hash
name-hash.pro
Headers
Sources
lib/StanfordCPPLib
src
name_hash.cpp
Other files

```
#include "console.h"
#include "simpio.h" // for getLine
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    last = getLine("What is your last name? ");
    value = nameHash(first, last);
    cout << "The hash of your name is: " << hashValue << endl;
}

/* This is the actual implementation of the nameHash function.
 * It takes two strings, first and last, and returns their hash value.
 * For those who are curious, it treats each character as a byte and
 * then uses a simple rolling hash algorithm.
 * Finally, it converts the hash value to a signed integer.
 */


```

Open Documents main.cpp name_hash.cpp

name hash

Threads: #1 name-hash Stopped: "function-finished".

Level Function File Line

0	Main	name_hash...	31
1	Main	main.cpp	23
2	startupMain	platform.cpp	2208
3	main	name_hash...	27

Number Function File Line Address Condition Ignore Threads

Name Value

first	@0x7fffffff0ao
last	@0x7fffffff0eoC0
hashValue	-590633613

returned value 1967457

We can see that the nameHash function returned 1967457. Thanks, debugger!



File Edit Build Debug Analyze Tools Window Help

Projects name-hash nameHash(string, string): int

#include <conio.h>
#include "simpio.h" /> Line
using namespace std;

/* Prototype for
 * in main and the
int nameHash(str

int main() {
 string first
 string last = "

int hashValue = nameHash(first, last);

cout << "The hash of your name is: " << hashValue << endl;

return 0;

}

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128
 * It then uses them as coefficients in a polynomial over the finite field
 * F_n where n is a large prime number and evaluates that polynomial at

Threads: #1 name-hash Stopped: "function-finished".

Views

Level Function File Line Number Function File Line Address Condition Ignore Threads

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages

But if you look up over here in the values window, you can see that **hashValue** has some really weird-looking number stored in it. (You'll almost certainly see something different on your system.)

returned value 1967457



#include <conio.h>
#include "simpio.h" />
using namespace std;

/* Prototype for
 * in main and the
int nameHash(
string first,
string last);

int main() {
 string first;
 string last = "";

int hashValue = nameHash(first, last);

cout << "The hash of your name is: " << hashValue << endl;
 return 0;
}

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128
 * It then uses them as coefficients in a polynomial over the finite field
 * F_n where n is a large prime number and evaluates that polynomial at

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Type to locate (Ctrl+F) Issues Search Results Application Output Compile Output QML/JS Console General Messages

1 2 3 4 5 6

Views

returned value 1967457



This is pretty cool, actually!

```
#include <console.h>
#include "simpio.h"
using namespace std;

/* Prototype for
 * in main and the
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128
 * It then uses them as coefficients in a polynomial over the finite field
 * F_n where n is a large prime number and evaluates that polynomial at
```

returned value 1967457

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Type to locate (Ctrl...) Issues Search Results Application Output Compile Output QML/JS Console General Messages



#include <conio.h>
#include "simpio.h" /> Line
using namespace std;

/* Prototype for
 * in main and the

int nameHash(string first, string last);

int main() {
 string first = "John";
 string last = "Doe";
 int hashValue = nameHash(first, last);
 cout << "The hash of your name is: " << hashValue << endl;

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128
 * It then uses them as coefficients in a polynomial over the finite field
 * F_n where n is a large prime number and evaluates that polynomial at
 * x = 256.

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Issues Search Results Application Output Compile Output QML/JS Console General Messages

What's happened is that we've just returned from nameHash with a value, but since we're going through the program one step at a time, we haven't actually assigned that value to hashValue yet!

returned value 1967457



#include <conio.h>
#include "simpio.h" /> Line
using namespace std;

/* Prototype for
 * in main and the
int nameHash(str
int main() {
 string first = getLine("What is your first name? ");
 string last = getLine("What is your last name? ");

 int hashValue = nameHash(first, last);

 cout << "The hash of your name is: " << hashValue << endl;
 return 0;
}

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128
 * It then uses them as coefficients in a polynomial over the finite field
 * F_n where n is a large prime number and evaluates that polynomial at

Threads: #1 name-hash Stopped: "function-finished".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	31								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

returned value 1967457

Views

Type to locate (Ctrl+F)

Issues Search Results Application Output Compile Output QML/JS Console General Messages



Line: 33, Col: 5

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	1967457	int

• you should see the right value get stored
(notice it's in red!) and we've moved to the
next line.

```
using namespace std;

/* Prototype for the nameHash function
 * in main and the nameHash function
 */
int nameHash(string first, string last);

int main() {
    string first;
    string last = getLine("What is your last name? ");

    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the actual function that computes the hash code. We're going
 * to talk more about what hash functions do later in the quarter. In
 * the meantime, think of it as a function that scrambles up the characters
 * of the input and produces a number.
 *
 * For those of you who are more mathematically inclined, this function
 * treats each character in the input name as a number between 0 and 128
 * It then uses them as coefficients in a polynomial over the finite field
 * F_p, where p is a large prime number, and evaluates that polynomial at
 * some smaller prime number q. (You aren't expected to know this for CS
 * but we thought it might be fun!)
 */

Threads: #1 name-hash Stopped: "end-stepping-range".
Views
```

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	33								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Type to locate (Ctrl+F)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages



File Edit Build Debug Analyze Tools Window Help

Projects name-hash

- name-hash.pro
- Headers
- Sources
 - lib/StanfordCPPLib
 - src
- Other files

20 using namespace std;

21

22 /* Prototype for the nameHash function, which will be defined later in the file.

23 * In main and the nameHash function, we'll use this prototype to declare the function.

24 */

25 int nameHash(string first, string last);

26

27 int main() {

28 string first; // Variable to store the first name.

29 string last; // Variable to store the last name.

30

31 int hashValue = nameHash(first, last);

32

33 cout << "The hash of your name is: " << hashValue << endl;

34 return 0;

35 }

36

37 /* This is the actual function that computes the hash code. We're going

38 * to talk more about what hash functions do later in the quarter. In

39 * the meantime, think of it as a function that scrambles up the character

40 * of the input and produces a number.

41 *

42 * For those of you who are more mathematically inclined, this function

43 * treats each character in the input name as a number between 0 and 128

44 * It then uses them as coefficients in a polynomial over the finite field

45 * \mathbb{F}_p , where p is a large prime number, and evaluates that polynomial at

46 * some smaller prime number q . (You aren't expected to know this for CS

47 * but we thought it might be fun!)

Line: 33, Col: 5

Name	Value	Type
first	@0x7fffffff0a0	std::string
last	@0x7fffffff0c0	std::string
hashValue	1967457	int

Threads: #1 name-hash Stopped: "end-stepping-range".

Level	Function	File	Line	Number	Function	File	Line	Address	Condition	Ignore	Threads
0	Main	name_hash...	33								
1	Main	main.cpp	23								
2	startupMain	platform.cpp	2208								
3	main	name_hash...	27								

Type to locate (Ctrl+F)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML/JS Console 6 General Messages

To do this, click on this button. If you hover over it, it says "Continue," and that button means "unpause the program and let it keep running from here."

```
using namespace std;

/* Prototype for the nameHash function. This lets us use the function
 * in main and then define it later in the program.
 */
int nameHash(string first, string last);

int main() {
    string first = getLine("What is your first name? ");
    string last = getLine("What is your last name? ");

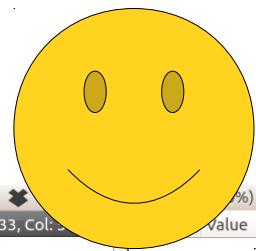
    int hashValue = nameHash(first, last);

    cout << "The hash of your name is: " << hashValue << endl;
    return 0;
}

/* This is the actual function that computes the hash. We're going to talk more about what hash functions do later. In the meantime, think of it as a function that takes two strings of the input and produces a number.
 */


```

The screenshot shows the Qt Creator IDE interface. On the left is a vertical toolbar with various icons for file operations, projects, and analysis. The central area displays a C++ code editor with the file 'name_hash.cpp' open. The code defines a function 'nameHash' and its main entry point. A callout bubble with handwritten text points to the toolbar at the bottom left, specifically to the button that typically has a green play/pause icon. To the right of the code editor is a status bar showing the line and column numbers (e.g., '# Line: 33, Col: 5'), the current time (12:00 PM), and a zoom level (100%). Below the code editor is a 'Views' dock containing tabs for Function, File, Line, Address, Condition, Ignore, and Threads. At the very bottom is a navigation bar with tabs for Issues, Search Results, Application Output, Compile Output, QML/JS Console, and General Messages.



If you do, you should see something like this.
(The program window might not automatically
pop up. That's okay! Just open it manually.)

Our program is now done running!

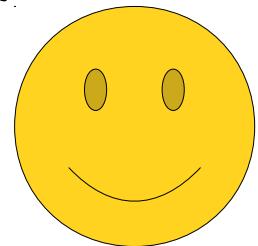
The screenshot shows the Qt Creator IDE interface. On the left is the project tree for 'name-hash' containing files like 'name-hash.pro', 'Headers', 'Sources', 'lib/StanfordCPPLib', 'src', and 'Other files'. Below that is the 'Open Documents' list with 'main.cpp' and 'name_hash.cpp'. The main area shows code in 'main.cpp' with a callout pointing to the terminal window. The terminal window title is 'Console' and contains the following text:

```
What is your first name? Ada
What is your last name? Lovelace
The hash of your name is: 1967457
```

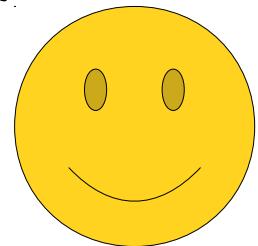
The code in 'main.cpp' includes a function definition for calculating a hash based on the names.

At the bottom, the status bar shows 'Debugger finished.' and the 'Threads' view with two columns: Level, Function, File, Line and Number, Function, File, Line, Address, Condition, Ignore, Threads.

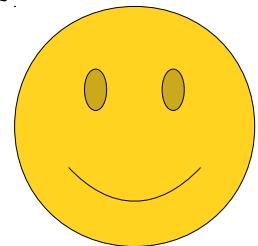
So there you have it! You've now gotten more
familiar with the debugger!



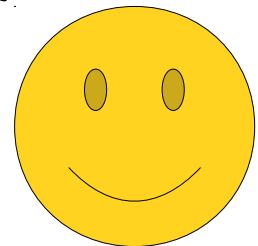
You know how to set a breakpoint to pause the program at a particular point.



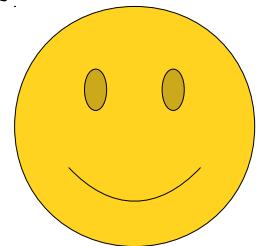
You know how to read the call stack and to see the values of local variables.



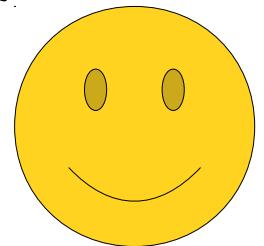
You know how to single-step the program and
see what values change.



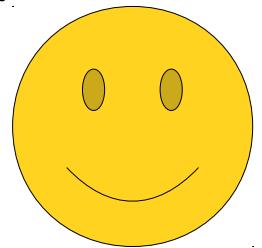
You know how to run a function to completion,
and how to let the program keep on running.



As you write more and more complicated programs this quarter, you'll get a lot more familiar using the debugger and seeing how your programs work.



And, if you continue to build larger and larger pieces of software, you'll find that knowing how to use a debugger is a surprisingly valuable skill!



Hope this helps, and welcome to CS106B!

