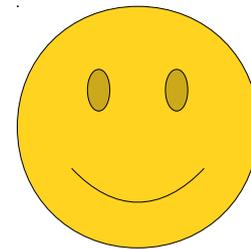
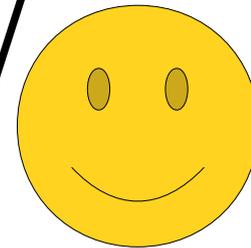


Assignment 0: Using the Debugger

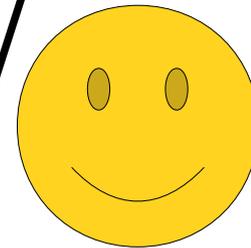
Hi everybody!



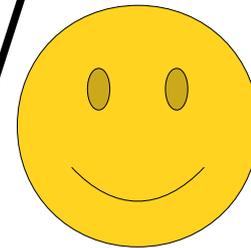
As part of Assignment 0, we'd like you to get a little bit of practice using the debugger in Qt Creator.



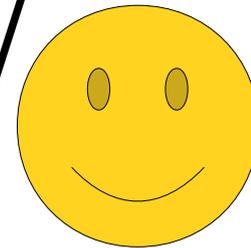
The debugger is a tool you can use to help see what your program is doing as you run it.



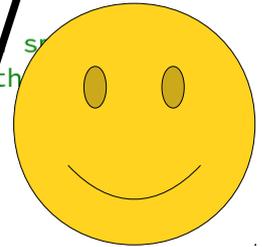
It's really useful for helping find errors in your programs, and the more practice you get with it, the easier it'll be to correct mistakes in the programs you write.



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.



```
42 * For t
43 * treat
44 * It th
45 * F_p, wh
46 * some smaller prime number q (you aren't expected to use q),
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 p and a smaller
51     * prime. These numbers were chosen because their product is less than 2^31.
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
```

Type to locate (Ctrl...

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

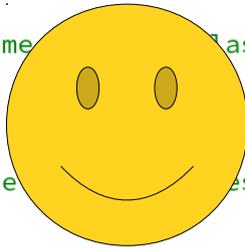
```
42 * For those of you who are more mathematically inclined, this function
43 * treats each character in the input name as a number between 0 and 128.
44 * It then uses them as coefficients in a polynomial over the finite field
45 *  $F_p$ , where  $p$  is a large prime number, and evaluates that polynomial at
```

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

Now, click the mouse!



```
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first name and 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 ASCII 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
```



Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

NameHash

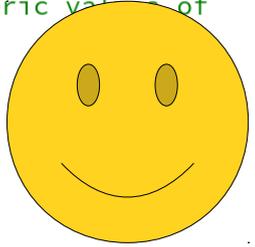
Debug

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

```
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 * GF(256) with modulus polynomial at CS106B,
60
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case. The numeric value 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
```

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.



Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

NameHash

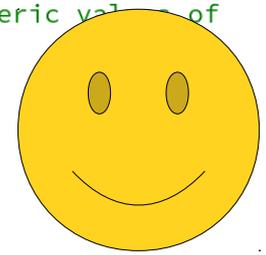
Debug

Run

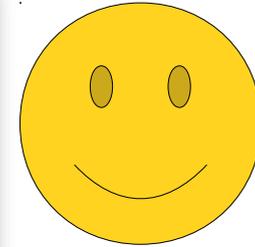
Run and Debug

```
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     * prime number q. (You aren't expected to know this for CS106B, but we
52     * thought it might be fun!) */
53     int hashVal = 0;
54     for (char ch: first + last) {
55         /* Convert the input character to lower case. The numeric value of
56         * lower-case letters are always less than 127. */
57         ch = tolower(ch);
58         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
59     }
60     return hashVal;
61 }
62
63
64
65
66
67
68
69
70
71
```

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.

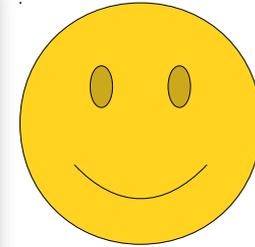


Qt Creator interface showing the NameHash project. The main editor displays C++ code with a red cursor at line 66. A console window titled "NameHash Console" is open, showing the prompt "What is your first name? |". The debugger window at the bottom shows the application has started.

```
45
46 * some smaller prime number q. (You aren
47 * but we thought it might be fun!)
48 */
49
50
51
52 What is your first name? |
53
54
55
56
57
58
59
60
61
62
63
64
65
66 ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kL
68 }
69 return hashVal;
```

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
					1eHash.cpp	66	...5555b6782			(all)

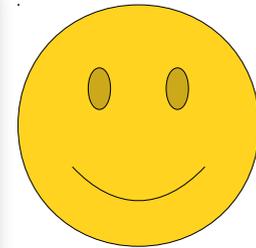
In the meantime, type in the first name **Ada** and hit enter, as shown here. We specifically want you to enter **Ada** here, *not your actual first name*.
(Unless your first name is Ada.)



```
45  
46 * some smaller prime number q. (You aren't  
47 * but we thought it might be fun!)  
48 */  
49  
50  
51  
52  
53 What is your first name? Ada  
54 What is your last name? |  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66 ch = tolower(ch);  
67 hashVal = (kSmallPrime * hashVal + ch) % kL  
68 }  
69 return hashVal;
```

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
					1eHash.cpp	66	...5555b6782			(all)

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



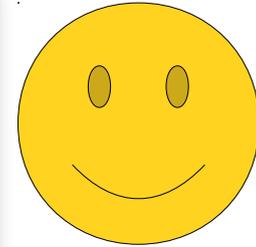
```
46 * some smaller prime number q. (You aren  
47 * but we thought it might be fun!)  
48 */  
49
```

```
52 What is your first name? Ada  
53  
54 What is your last name? Lovelace  
55
```

```
66 ch = tolower(ch);  
67 hashVal = (kSmallPrime * hashVal + ch) % kL  
68 }  
69 return hashVal;
```

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
					1eHash.cpp	66	...5555b6782			(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.



```
46  * some smaller prime number q. (You aren  
47  * but we thought it might be fun!)  
48  */
```

```
52  What is your first name? Ada  
53  
54  What is your last name? Lovelace
```

```
66  ch = tolower(ch);  
67  hashVal = (kSmallPrime * hashVal + ch) % kL  
68  }  
69  return hashVal;
```

Debugger → GDB for "NameHash" | Threads: #12 | Application started.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
					1eHash.cpp	66	...5555b6782			(all)

Type to locate (Ctrl...

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

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

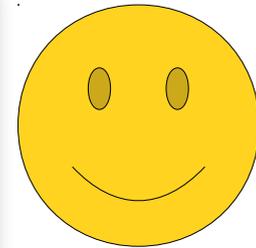
```
46 * some smaller prime number q. (You aren  
47 * but we thought it might be fun!)  
48 */
```

```
52 What is your first name? Ada  
53  
54 What is your last name? Lovelace
```

```
66 ch = tolower(ch);  
67 hashVal = (kSmallPrime * hashVal + ch) % kL  
68 }  
69 return hashVal;
```

Debugger → GDB for "NameHash" | Threads: #12 | Application started.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
					1eHash.cpp	66	...5555b6782			(all)



Activities NameHash

File Edit View Build Debug Analyze Tools Window Help

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

NameHash Console

File Edit Options Help

Debugger → GDB for "NameHash" | Threads: #12 | Application started.

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

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

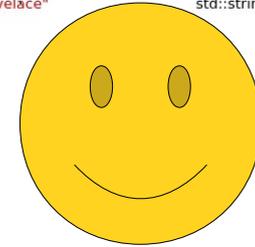
The screenshot shows the Qt Creator IDE with the following components:

- Code Editor:** A C++ function `int nameHash(string first, string last)` is shown. It includes comments about a hashing scheme using two prime numbers, `kLargePrime = 16908799` and `kSmallPrime = 127`. The function iterates over the concatenated string `first + last`, converting characters to lowercase and calculating a hash value.
- Variable Inspector:** A window on the right shows the state of variables. It lists `std::string::iterator`, `char`, `int`, and `std::string` with their respective values and addresses. A yellow smiley face is drawn over this window.
- Debugger Console:** At the bottom, the debugger shows the execution stack. It is stopped at breakpoint 1 in thread 12, at line 66 of `NameHash.cpp`. The stack includes `nameHash`, `studentMain`, and `start thread`.

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

```
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers
51     * prime. These numbers were chosen because they are
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
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case
64         * lower-case letters are always less than
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

```
"AdaLovelace"
'A' 65 0x41
"Ada"
0
16908799
127
"Lovelace"
```



Name	Value	Type
		std::string::iterator
		std::string::iterator
		std::string &&
		char
		std::string
		int
		int
		int
		std::string

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

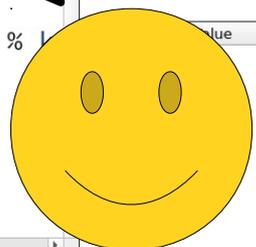
Build

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers:
51     * prime. These numbers were chosen because the
52     *  $2^{31} - kLargePrime - 1$ .
```

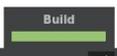
Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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

```
62 for (char ch: first + last)
63     /* Convert the input character to lowercase.
64     * lower-case letters are always
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```



Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								



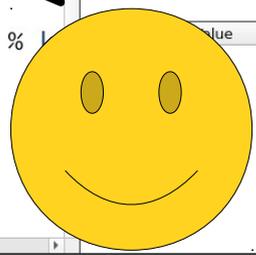
```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers;
51        * prime. These numbers were chosen because the
52        *  $2^{31} - kLargePrime - 1$ .
```

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65	char 0x41
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

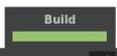
This yellow arrow indicates where in the program we are right now. The program stopped running at this line because we hit that breakpoint you set earlier.



```
62 for (char ch: first + last)
63     /* Convert the input character to lower case.
64        * lower-case letters are always
65        */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```



Level	Function	File	Line	Address	Number	Funcnt File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x5555555b6782	1	...g) ...eHash.cpp	66	...55555b6782			(all)
2	studentMain	NameHash.cpp	31	0x5555555b6595							
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x55555556161bc							
4	GThreadStd::run()			0x5555555f9476							
5	??			0x7ffff6143d84							
6	start thread	pthread_create.c	463	0x7ffff6257590							



Activities Qt Creator Jan 4 3:15 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers;
51        * prime. These numbers were chosen because the
52        * 2^31 - kLargePrime - 1.
```

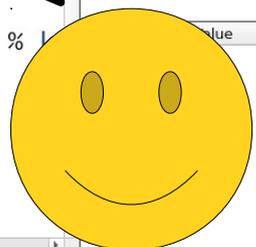
Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x5555555b6782	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x5555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x55555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

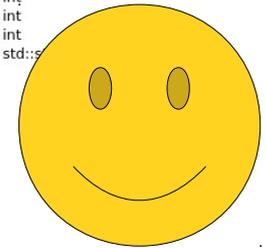
Build

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.



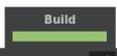
```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first
60  * name, updating the hash at each step.
61  */
62  for (char ch : first)
63  {
64  *
65  */
66  ch
67  hash
68  }
69  return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



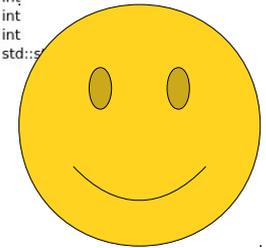
Next, let's take a look at this panel.
This is called the **call stack**.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								



```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::s



Right now, we know we're in the nameHash function, because our helpful friend the Yellow Arrow tells us exactly what line we're on!

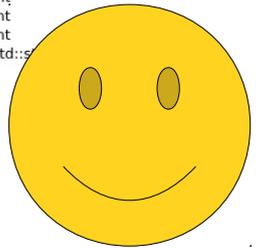


Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::s



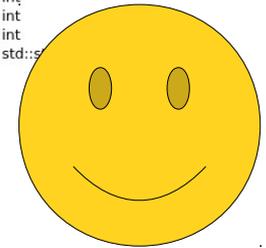
However, the yellow arrow can't tell us exactly how we got to this part of the program. What part of the program actually called nameHash?

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

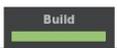
```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal = 0;
58
59  /* Iterate across all the characters in the first
60  * name, updating the hash at each step.
61  */
62  for (char ch : first)
63  /*
64  *
65  */
66  {
67  hashVal = (hashVal * kLargePrime + ch) % kSmallPrime;
68  }
69  return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"Ada Lovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



The call stack can tell us exactly that!

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								



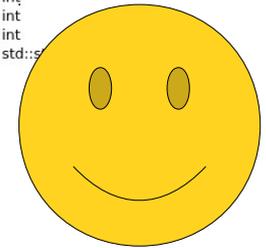
Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::s



Go and double-click the call to studentMain on Level 2. When you do...

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	NameHash	NameHash.cpp	66	0x555555b6782	1				(all)
2	studentMain	NameHash.cpp	31	0x5555555b6595								
3	std::thread::handle_int()	QGuiEventBackendEv		0x5555555161b								
4	GThreadSt			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start threa	pthread create.c	463	0x7ffff6257590								

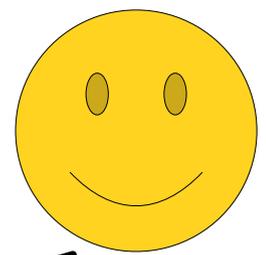


Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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: " << hashVal
34     return 0;
35 }
36
37 /* This is the actual function that c
38 * to talk more about what hash funct
39 * the meantime, think of it as a func
40 * of the input and produces a number
41 *
42 * For those of you who are more mathem
43 * treats each character in the input name as a num
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovellace"	std::string



You'll end up over here!

Debugger

GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x5555555b6782								
2	studentMain	NameHash.cpp	31	0x5555555b6595	1eHash.cpp	66	...5555555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Activities Qt Creator Jan 4 3:22 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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: " << hashVal
34     return 0;
35 }
36
37 /* This is the actual function that c
38 * to talk more about what hash funct
39 * the meantime, think of it as a fun
40 * of the input and produces a number
41 *
42 * For those of you who are more math
43 * treats each character in the input
```

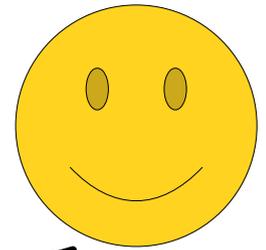
Debugger GDB for "NameHash" Threads: #12 Na

Level	Function	File	Line	Addr
1	nameHash	NameHash.cpp	66	0x55
2	studentMain	NameHash.cpp	31	0x55
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x55
4	GThreadStd::run()			0x55
5	??			0x7ff
6	start thread	pthread create.c	463	0x7ff

Name Value Type

first	"Ada"	std::string
hashValue	0	int
last	"Lovellace"	std::string

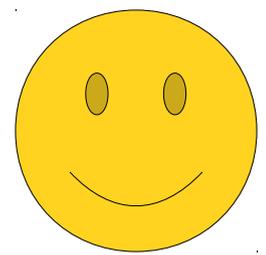
Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Con



Notice that the yellow arrow points to Line 31. That line 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!

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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: " << hashVal
34     return 0;
35 }
36
37 /* This is th
38 * to talk mo
39 * the meanti
40 * of the inp
41 *
42 * For those
43 * treats each
```

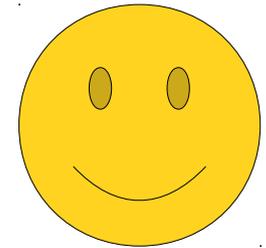
Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovellace"	std::string



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!

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Name	Value	Type
first	"Ada"	std::string
hashCode	0	int
last	"Lovelace"	std::string



Depending on your OS, you might see some additional functions beneath studentMain. What are those?

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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 <<
34     return
35 }
36
37 /* This is
38 * to talk
39 * the meant
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inc
43 * treats each character in the input name as a num
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovelace"	std::string



These grayed-out functions represent helper functions our libraries automatically call to help get your program set up.

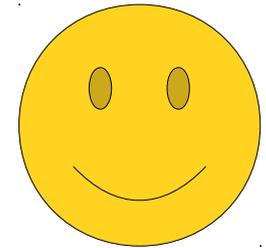
Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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 <<
34     return
35 }
36
37 /* This is
38 * to talk
39 * the meant
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inc
43 * treats each character in the input name as a num
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovelace"	std::string



You don't need to worry about these. They'll show up in all the programs you run and you can safely ignore them.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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 <<
34     return
35 }
36
37 /* This is
38 * to talk
39 * the meant
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inc
43 * treats each character in the input name as a num
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovellace"	std::string



In the meantime, let's get back to our nameHash function. To do that, double-click on the nameHash entry at the top of the call stack. When you do...

Level	Function	File	Line	Address	Number	Funcnt	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x5555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	std::t...	...	3	0x...								
3	std::_Func...	handler<int ()	QtGui::startBackgroundEve...	0x5555556161bc								
4	GThreadSt			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start threa	pthread create.c	463	0x7ffff6257590								

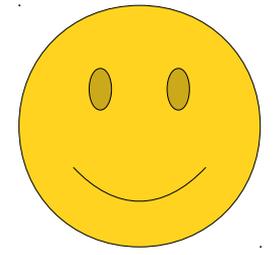
Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64  * lower-case letters are always less than
65  */
66  ch
67  has
68  }
69  return
70  }
71
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



You'll be teleported back here!

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

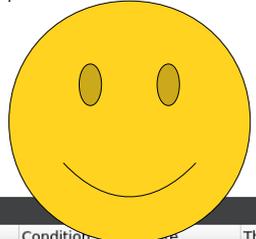
```

48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51   * prime. These numbers were chosen because the
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  /* I
60   * r
61   */
62  for
63
64
65
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71

```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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



Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	View	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

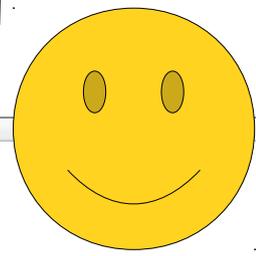
Projects

Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65 0x41	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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.



Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

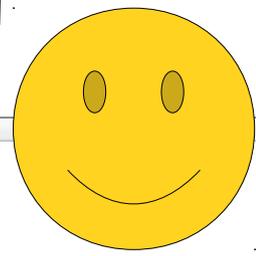
NameHash

Debug

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers:
51  * prime. These numbers were chosen because the
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
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71
```

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65 0x41	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Once the breakpoint is reached, it will pull up all sorts of useful information.



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

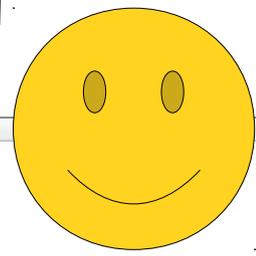
Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x5555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x5555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x55555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers;
51     * prime. These numbers were chosen because the
52     * 2^31 - kLargePrime - 1.
53     */
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65 0x41	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

The yellow arrow points out where we are right now.

```
62 for (char ch: first + last) {
63     /* Convert the input character to lower case
64     * lower-case letters are always less than
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kL
68 }
69 return hashVal;
70 }
71
```



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

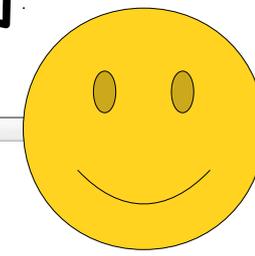
NameHash

Debug

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers:
51  * prime. These numbers were chosen because the
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
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71
```

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65 0x41	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), Qt...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

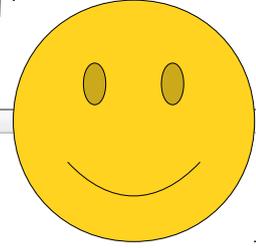
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71
```

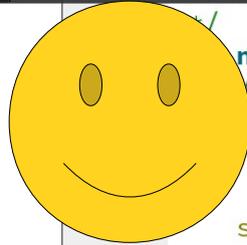
Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"A" 65 0x41	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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



Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Look up at this panel over here.



```
nameHash(string first, string last){  
    /* This hashing scheme needs two prime numbers:  
    * prime. These numbers were chosen because the  
    *  $2^{31} - kLargePrime - 1$ .  
    */  
    static const int kLargePrime = 16908799;  
    static const int kSmallPrime = 127;  
  
    int hashVal = 0;  
  
    /* Iterate across all the characters in the first  
    * name, updating the hash at each step.  
    */  
    for (char ch: first + last) {  
        /* Convert the input character to lower case  
        * lower-case letters are always less than  
        */  
        ch = tolower(ch);  
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;  
    }  
    return hashVal;  
}
```

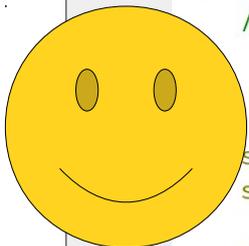
Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (<int >), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Type to locate (Ctrl...

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

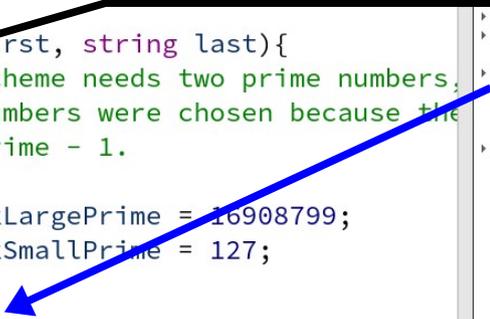
We can also see that, at this point, hashVal is still zero.



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because they
    * are the two largest primes 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, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case;
        * lower-case letters are always less than
        * upper-case letters.
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
```



Name	Value	Type
__for_end	@0x7fff6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string::iterator
ch	'A'	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Activities Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects

NameHash [main]

NameHash.pro

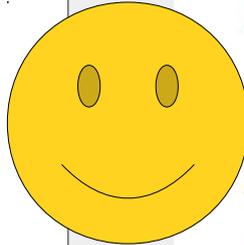
Sources

NameHash.cpp

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

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



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

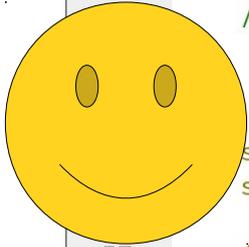
    int hashVal = 0;

    57
    58
    59  /* Iterate across all the characters in the first
    60  * name, updating the hash at each step.
    61  */
    62  for (char ch: first + last) {
    63  /* Convert the input character to lower case
    64  * lower-case letters are always less than
    65  */
    66  ch = tolower(ch);
    67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    68  }
    69  return hashVal;
    70  }
    71
```

Name	Value	Type
__for_end	@0x7fff6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string&&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

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



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers.
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

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

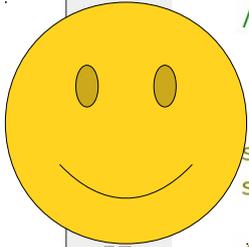
Debugger variable view for `nameHash` at line 66:

Name	Value	Type
<code>__for_end</code>	<code>@0x7fff6058c80</code>	<code>std::string::iterator</code>
<code>__for_range</code>	<code>"AdaLovelace"</code>	<code>std::string &&</code>
<code>ch</code>	<code>'A'</code>	<code>char</code>
<code>first</code>	<code>"Ada"</code>	<code>std::string</code>
<code>hashVal</code>	<code>0</code>	<code>int</code>
<code>kLargePrime</code>	<code>16908799</code>	<code>int</code>
<code>kSmallPrime</code>	<code>127</code>	<code>int</code>
<code>last</code>	<code>"Lovelace"</code>	<code>std::string</code>

Debugger thread view for `GDB for "NameHash"`:

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	<code>nameHash</code>	<code>NameHash.cpp</code>	<code>66</code>	<code>0x555555b6782</code>	1		<code>...g) ...eHash.cpp</code>	<code>66</code>	<code>...5555b6782</code>			(all)
2	<code>studentMain</code>	<code>NameHash.cpp</code>	<code>31</code>	<code>0x555555b6595</code>								
3	<code>std::_Function_handler<int (), QtGui::startBackgroundEve...</code>			<code>0x5555556161bc</code>								
4	<code>GThreadStd::run()</code>			<code>0x5555555f9476</code>								
5	<code>??</code>			<code>0x7ffff6143d84</code>								
6	<code>start thread</code>	<code>pthread_create.c</code>	<code>463</code>	<code>0x7ffff6257590</code>								

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



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

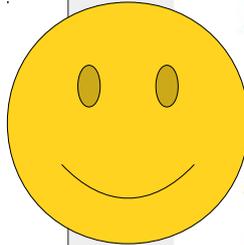
    int hashVal = 0;

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

Name	Value	Type
__for_end	@0x7fff6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

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.



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

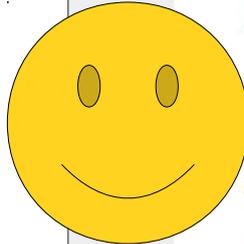
    /* Iterate across all the characters in the first
    * name, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case;
        * lower-case letters are always less than
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime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

```
type
std::string::iterator
std::string::iterator
std::string &&
char
std::string
int
int
int
std::string
std::string::iterator @0x7fff6058c80
for range "Ada Lovelace"
ch "A" 65 0x41
hashVal 0
kLargePrime 16908799
kSmallPrime 127
last "Lovelace"
```

Name	Value	Type
------	-------	------

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

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.



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

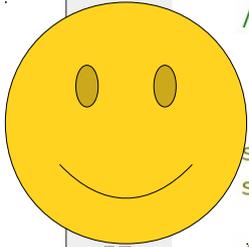
    int hashVal = 0;

    /* Iterate across all the characters in the first
    * name, updating the hash at each step.
    */
    for (char ch: first + last) {
        /* Convert the input character to lower case;
        * lower-case letters are always less than
        */
        ch = tolower(ch);
        hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
    }
    return hashVal;
}
70 }
71
```

Name	Value	Type
__for_end	@0x7fff6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string&&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

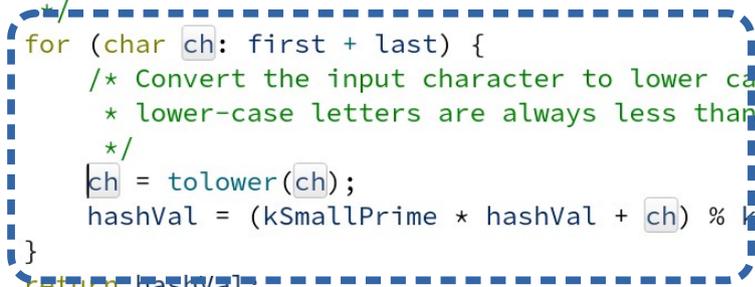
Now, let's do something really cool - we're going to run this program one line at a time, watching what happens at each step!



```
48  */
49  int nameHash(const string& first, string last){
    /* This hashing scheme needs two prime numbers,
    * prime. These numbers were chosen because they
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

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

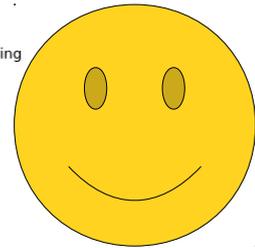


Name	Value	Type
__for_end	@0x7fff6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string::iterator
ch	'A'	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int ()>, QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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

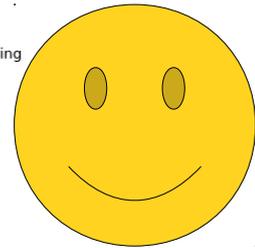


Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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



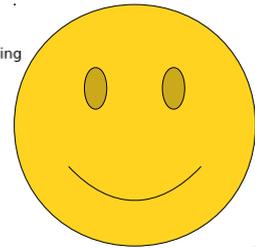
Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782								
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782			(all)
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```

48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51   * prime. These numbers were chosen because the
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 fir
60   * name, updating the hash at each step.
61   */
62  for (ch
63  /*
64   *
65   */
66  ch
67  has
68  }
69  return hashVal;
70  }
71

```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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."



Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'A' 65	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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...

Debugger GDB for "NameHash" Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBa...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								



Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Debugger

Level Function

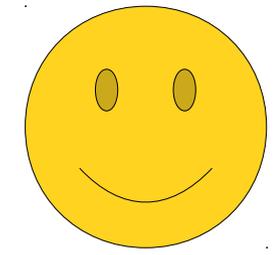
- 1 nameHash
- 2 studentMain
- 3 std::_Function_handler<int (), Q
- 4 GThreadStd::run()
- 5 ??
- 6 start thread

```

48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71

```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



...your window should look something like this.

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Debugger GDB for "NameHash"

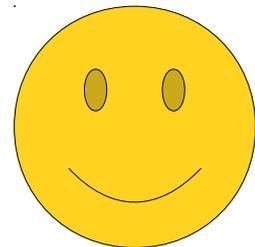
Level	Function
1	nameHash
2	studentMain
3	std::_Function_handler<int (), Q
4	GThreadStd::run()
5	??
6	start thread

Type to locate (Ctrl...

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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

Projects

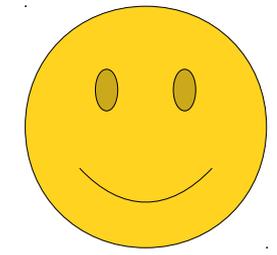
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Debugger

- Level Function
- 1 nameHash
- 2 studentMain
- 3 std::Function_handler<int (), Q
- 4 GThreadStd::run()
- 5 ??
- 6 start thread

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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

Projects

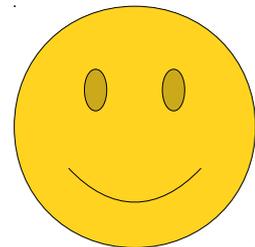
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Debugger

Level	Function
1	nameHash
2	studentMain
3	std::_Function_handler<int (), Q
4	GThreadStd::run()
5	??
6	start thread

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % k
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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

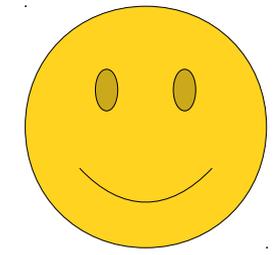
Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case;
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



so what did that line of code do?

Debugger GDB for "NameHash"

Level	Function
1	nameHash
2	studentMain
3	std::_Function_handler<int (), QThreadStd::run()
4	GThreadStd::run()
5	??
6	start thread

Projects

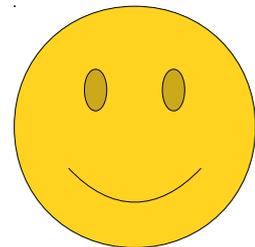
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

NameHash Debug

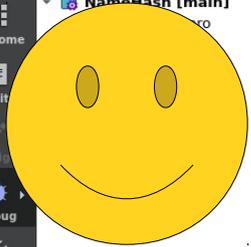
```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 127;
56
57  int hashVal =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

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



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!



```
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

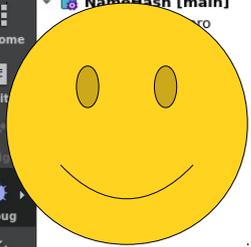
    int hashVal = 0;

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

	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start_thread	pthread_create.c	463	0x7ffff6257590								

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

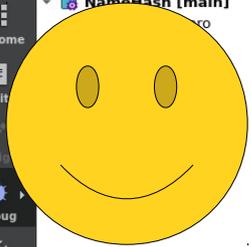


```
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers;
51     * prime. These numbers were chosen because the
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54     static const int kLargePrime = 16908799;
55     static const int kSmallPrime = 127;
56
57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

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	Ada Lovelace	std::string &&
ch	'a'	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66		(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

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



```
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers;
51     * prime. These numbers were chosen because the
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
60     * name, updating the hash at each step.
61     */
62     for (char ch: first + last) {
63         /* Convert the input character to lower case
64         * lower-case letters are always less than
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68     }
69     return hashVal;
70 }
71 }
```

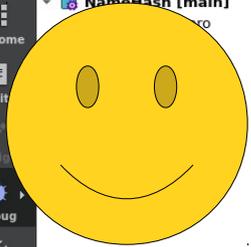
	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	Ada Lovelace	std::string &&
ch	a	char
first	Ada	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Type to locate (Ctrl...

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

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.



```
int nameHash(string first, string last){
    /* This hashing scheme needs two prime numbers;
    * prime. These numbers were chosen because the
    *  $2^{31} - kLargePrime - 1$ .
    */
    static const int kLargePrime = 16908799;
    static const int kSmallPrime = 127;

    int hashVal = 0;

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

	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	Ada Lovelace	std::string &&
ch	'a'	char
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

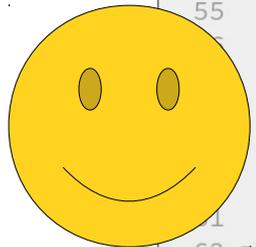
Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start_thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

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



```
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
```

```
int hashVal = 0;

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

Type

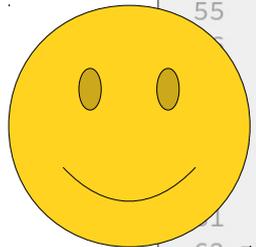
- ::string::iterator
- ::string::iterator
- ::string &&
- char
- ::string
- ::string

Name	Value	Type
------	-------	------

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

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.



```
54 start = first + last;
55 startFile = 127;
56 int hashVal = 0;
57
58 /* Iterate across all the characters in the first
59 * name, updating the hash at each step.
60 */
61
62 for (char ch: first + last) {
63     /* Convert the input character to lower case
64     * lower-case letters are always less than
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % k
68 }
69 return hashVal;
70 }
71
```

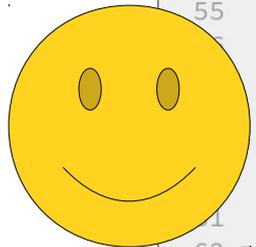
Type

- ::string::iterator
- ::string::iterator
- ::string &&
- ar
- ::string
- ::string

Name	Value	Type
------	-------	------

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

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

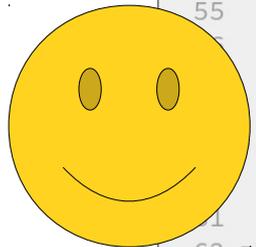


```
54 start = first + last;
55 startFile = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case
64     * lower-case letters are always less than
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kSmallPrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value	Type
		::string::iterator
		::string::iterator
		::string &&
		ar
		::string
		::string

Level	Function	File	Line	Address	Number	Funct File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1	...	66	...			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595							
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc							
4	GThreadStd::run()			0x5555555f9476							
5	??			0x7ffff6143d84							
6	start thread	pthread_create.c	463	0x7ffff6257590							

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



```
54 start = first + last;
55 startFile = 127;
56
57 int hashVal = 0;
58
59 /* Iterate across all the characters in the first
60 * name, updating the hash at each step.
61 */
62 for (char ch: first + last) {
63     /* Convert the input character to lower case
64     * lower-case letters are always less than
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % k
68 }
69 return hashVal;
70 }
71
```

Name	Value	Type
		::string::iterator
		::string::iterator
		::string &&
		ar
		::string
		::string

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
60  * name, updating the hash at each step.
61
62
63
64
65
66
67
68
69  return hashVal;
70 }
71
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	0	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



Hover over the "Step Over" button, confirm that the button you're clicking really is "Step Over," and click it! When you do...

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startB...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

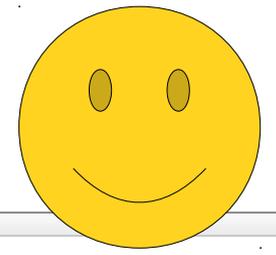
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Debugger

Level	Function
1	nameHash
2	studentMain
3	std::_Function_handler<int (), QtGui::startBackgroundEve...
4	GThreadStd::run()
5	??
6	start thread

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51   * prime. These numbers were chosen because the
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 fir
60   * name, updating the hash at each step.
61   */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64   * lower-case letters are always less than
65   */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"a" 97	char 0x61
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



... you'll end up with something like this!

Projects

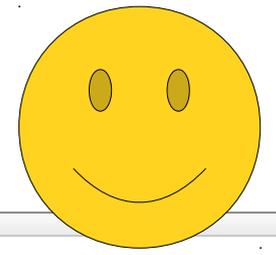
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Debugger

Level	Function
1	nameHash
2	studentMain
3	std::_Function_handler<int (), QtGui::startBackgroundEve...
4	GThreadStd::run()
5	??
6	start thread

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower cas
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"a" 97	char 0x61
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

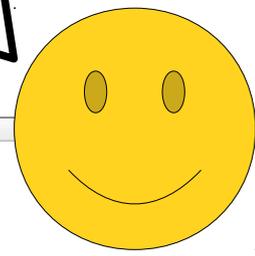


Let's see what's changed.


```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers:
51  * prime. These numbers were chosen because the
52  *  $2^{31} - kLargePrime$ 
53  */
54  static
55  static
56
57  int has
58
59  /* Iter
60  * name
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kL
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int

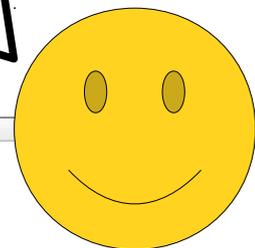
Second, notice that we're back up at the top of the for loop, since that's where the yellow arrow is pointing. We ended up back here because this is the next line that gets executed.



Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb	1eHash.cpp	66		(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'a' 97	char
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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

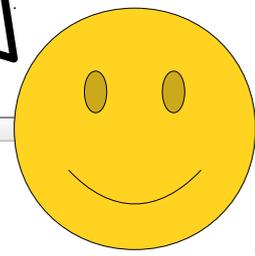


Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 over the characters in the strings
60  * nameFirst and nameLast.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case.
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	"a" 97	char 0x61
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Let's go do it again!

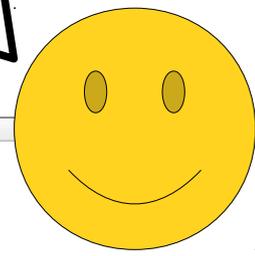


Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start_thread	pthread_create.c	463	0x7ffff6257590								

```
48 */
49 int nameHash(string first, string last){
50 /* This hashing scheme needs two prime numbers;
51 * prime. These numbers were chosen because the
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 over the characters in the string
60 * nameHash.
61 */
62 for (char ch: first + last) {
63 /* Convert the input character to lower case.
64 * lower-case letters are always less than
65 */
66 ch = tolower(ch);
67 hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"Ada Lovelace"	std::string &&
ch	'a' 97	char 0x61
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startB...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

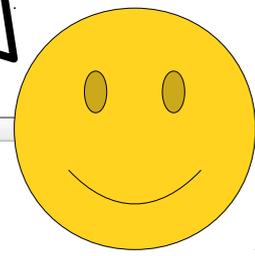
Projects

Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 over the characters in the string.
60  * nameHash will be called with first and last
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case.
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	Ada Lovelace	std::string &&
ch	d	char
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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



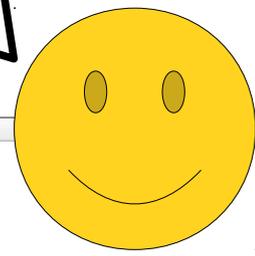
Debugger GDB for "NameHash" Threads: #12 NameHash Stopped at breakpoint 1 in thread 12.

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 over the characters in the string
60  * name.
61  */
62  for (char ch: first + last) {
63  /* Convert the input character to lower case.
64  * lower-case letters are always less than
65  */
66  ch = tolower(ch);
67  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68  }
69  return hashVal;
70 }
71
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"Ada Lovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Go click "Step Over" again to run this line of code.



Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	66	0x555555b6782	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startB...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

NameHash

Debug

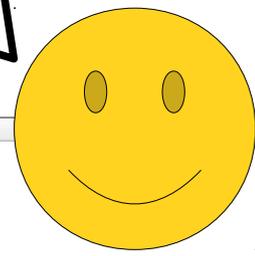
```

48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51   * prime. These numbers were chosen because the
52   * 2^31 - kLargePrime - 1.
53   */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 100;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the string.
60   * nameHash will be called with first and last
61   * parameters that define the range of the string.
62   */
63  for (char ch: first + last) {
64    /* Convert the input character to lower case.
65     * lower-case letters are always less than
66     * upper-case letters.
67     */
68    ch = tolower(ch);
69    hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
70  }
71  return hashVal;

```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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'.



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

Projects

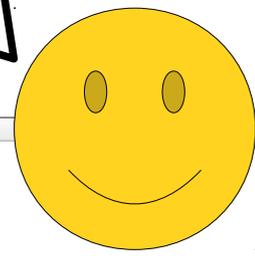
- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
52  * 2^31 - kLargePrime - 1.
53  */
54  static const int kLargePrime = 16908799;
55  static const int kSmallPrime = 100;
56
57  int hashVal = 0;
58
59  /* Iterate over the characters in the string.
60  * nameHash will be called with first and last
61  * parameters that define the range of characters
62  * to iterate over.
63  */
64  for (char ch: first + last) {
65  /* Convert the input character to lower case.
66  * lower-case letters are always less than
67  * upper-case letters.
68  */
69  ch = tolower(ch);
70  hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
71  }
72  return hashVal;
73 }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	97	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

Now, click "Step Over" one more time.



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

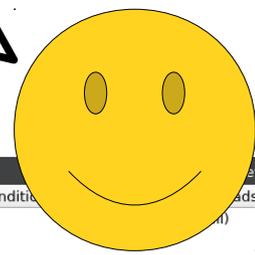
Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	67	0x555555b6790	1eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startB...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
52  *  $2^{31} - kLargePrime - 1$ .
53  */
54  static const int kLargePrime = 16908799;
```

Name	Value	Type
__for_begin	@0x7fffc6058c78	std::string::iterator
__for_end	@0x7fffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'd' 100	char
first	"Ada"	std::string &&
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string &&

Look here!

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!



```
60
61
62  }
63  return hashVal;
64 }
65
66
67
68
69
70
71
```

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition
1	nameHash	NameHash.cpp	62	0x555555b67cb						
2	studentMain	NameHash.cpp	31	0x555555b6595	1eHash.cpp	66	...5555b6782	
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc						
4	GThreadStd::run()			0x555555f9476						
5	??			0x7ffff6143d84						
6	start thread	pthread_create.c	463	0x7ffff6257590						

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers:
51     * prime. These numbers were chosen because the
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"Ada Lovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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.

```
61
62 for (char ch: first + last) {
63     /* Convert the input character to lower case
64     * lower-case letters are always less than
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```

Name	Value	Type
------	-------	------



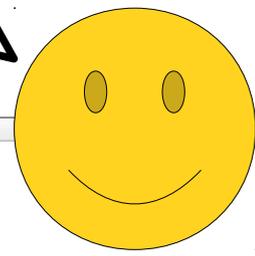
Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers:
51     * prime. These numbers were chosen because the
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"Ada Lovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

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

```
61
62 for (char ch: first + last) {
63     /* Convert the input character to lower case
64     * lower-case letters are always less than
65     */
66     ch = tolower(ch);
67     hashVal = (kSmallPrime * hashVal + ch) % kLargePrime;
68 }
69 return hashVal;
70 }
71
```



Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb	1		...g) ...eHash.cpp	66	...5555b6782			(all)
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start_thread	pthread_create.c	463	0x7ffff6257590								

Activities Qt Creator Jan 4 3:57 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
48 */
49 int nameHash(string first, string last){
50     /* This hashing scheme needs two prime numbers:
51     * prime. These numbers were chosen because the
52     *  $2^{31} - kLargePrime - 1$ .
53     */
54
61
62     for (char ch: first + last) {
63         /* Convert the input character to lower case
64         * lower-case letters are always less than
65         */
66         ch = tolower(ch);
67         hashVal = (kSmallPrime * hashVal + ch) % kL
68     }
69     return hashVal;
70 }
71
```

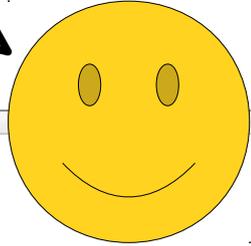
Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb								
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

NameHash Debug

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

... 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!

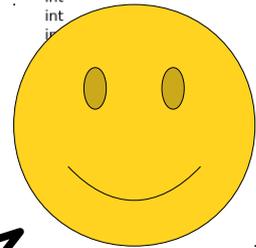


Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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 fir
60  * name, updating the hash at each step.
61  */
62  for (ch
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string



Now, take a look back at these buttons.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb								
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

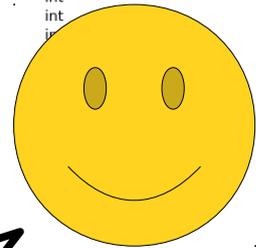
Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome Edit Design Debug Projects Help

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
60  * name, updating the hash at each step.
61  */
62  for (char ch : first)
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

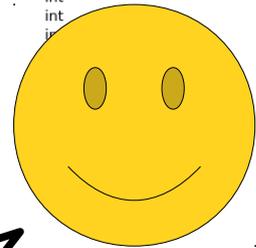


Hover your mouse over the one that's on the far right. When you hover over it, it should say "step out."

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb								
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
60  * name, updating the hash at each step.
61  */
62  for (char ch : first)
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70  }
71  }
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"AdaLovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

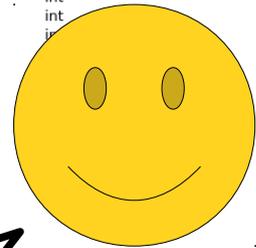


Don't click just yet. But when you do click, it will run the rest of the nameHash function until it finishes and returns.

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb								
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
4	GThreadStd::run()			0x555555f9476								
5	??			0x7ffff6143d84								
6	start_thread	pthread_create.c	463	0x7ffff6257590								

```
48  */
49  int nameHash(string first, string last){
50  /* This hashing scheme needs two prime numbers;
51  * prime. These numbers were chosen because the
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
60  * name, updating the hash at each step.
61  */
62  for (char ch : first)
63  /*
64  *
65  */
66  ch
67  has
68  }
69  return hashVal;
70 }
71
```

Name	Value	Type
__for_begin	@0x7ffc6058c78	std::string::iterator
__for_end	@0x7ffc6058c80	std::string::iterator
__for_range	"Ada Lovelace"	std::string &&
ch	'd' 100	char 0x64
first	"Ada"	std::string
hashVal	????	int
kLargePrime	16908799	int
kSmallPrime	127	int
last	"Lovelace"	std::string

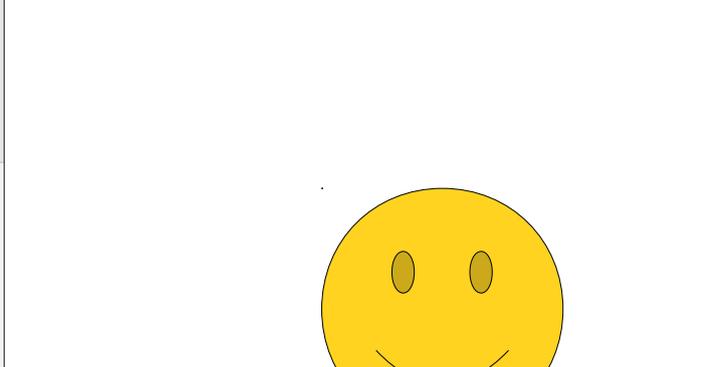


Now, go click that button. If you did everything right...

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	nameHash	NameHash.cpp	62	0x555555b67cb								
2	studentMain	NameHash.cpp	31	0x555555b6595								
3	std::_Function_handler<int (), QtGui::startBackground			0x5555556161bc								
4	GThreadStd::run()			0x5555555f9476								
5	??			0x7ffff6143d84								
6	start_thread	pthread_create.c	463	0x7ffff6257590								

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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: " << hashVal
34     return 0;
35 }
36
37 /* This is
38 * to talk
39 * the mean
40 * of the i
41 *
42 * For thos
43 * treats e
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovelace"	std::string



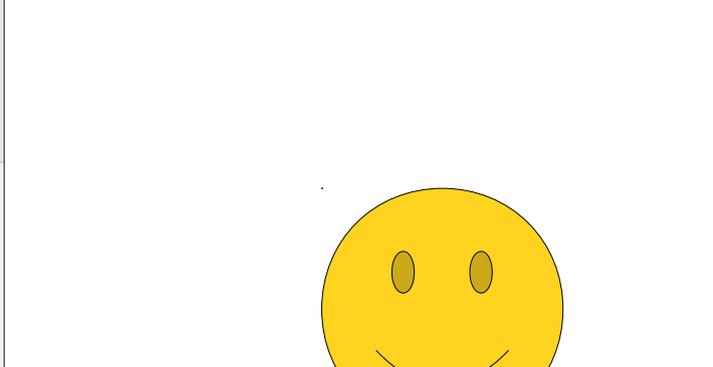
... you should end up with something that looks like this!

returned value 1967457 int

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x555555b6595								
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
3	GThreadStd::run()			0x5555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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: " << hashVal
34     return 0;
35 }
36
37 /* This is
38 * to talk
39 * the mean
40 * of the i
41 *
42 * For thos
43 * treats e
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovellace"	std::string



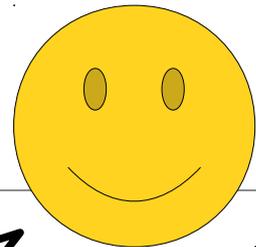
returned value 1967457 int

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

Level	Function	File	Line	Address	Number	Funcnt File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x555555b6595							
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc							
3	GThreadStd::run()			0x5555555f9476							
4	??			0x7ffff6143d84							
5	start_thread	pthread_create.c	463	0x7ffff6257590							
6	clone	clone.S	95	0x7ffff5e30223							

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovellace"	std::string

returned value 1967457 int



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

- Welcome
- Edit
- Design
- Debug
- Projects
- Help

NameHash Debug

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x555555b6595								
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
3	GThreadStd::run()			0x5555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								

Activities Qt Creator Jan 4 4:02 PM NameHash.cpp @ NameHash [main] - Qt Creator

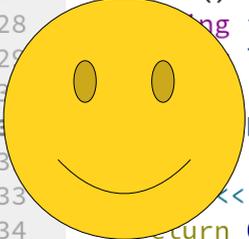
File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
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     last = getLine("What is your last name?
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashVal
34     return 0;
35 }
36
37 /* This is th
38 * to talk m
39 * the meant
40 * of the in
41 *
42 * For those
43 * treats ea
```

Name	Value	Type
first	"Ada"	std::string
hashValue	0	int
last	"Lovelace"	std::string

returned value 1967457 int



We can see that the nameHash function returned 1967457. Thanks, debugger!
(A note: it seems like on some Macs, this number doesn't display. Don't worry if you don't see it - just continue on as usual.)

Debugger GDB for "NameHash"

Level	Function
1	studentMain
2	std::_Function_handler<int (), QtG...
3	GThreadStd::run()
4	??
5	start_thread
6	clone

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

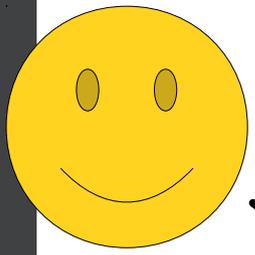
```
19 #include "simpio.h" // for getLine
20 using namespace std;
21
22 /* Prototype for the nameHash function. This lets u
23 * in main and then define it later in the program.
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first = getLine("What is your first name
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: " << hashVal
34     return 0;
35 }
36
37 /* This is the actual function that computes the ha
38 * to talk more about what hash functions do later
39 the meantime, think of it as a function that scr
40 the input and produces a number.
```

Name	Value	Type
hashValue	0	int

returned value 1967457 int

Name	Value	Type
------	-------	------

Line	Address	Condition	Ignore	Threads
------	---------	-----------	--------	---------



But if we look up over here, we see that hashValue isn't storing 1967457, even though that's what was returned.

(You might see a number other than 0 on your system - that's okay.)



Projects

- NameHash [main]
 - NameHash.pro
 - Sources
 - NameHash.cpp

Welcome

Edit

Design

Debug

Projects

Help

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and simpio.h
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
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 function that computes the hash value.
38 * to talk more about what hash functions do later
39 * the meantime, think of it as a function that scans
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined,
43 * treats each character in the input name as a number.
```

But it looks like we're setting hashValue equal to the number that was returned by the nameHash function. What's going on?

Name	Value	Type
first	"Ada"	std::string
returned value	1967457	int

Name	Value	Type
------	-------	------

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x555555b6595								
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
3	GThreadStd::run()			0x5555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								



Jan 4 4:02 PM

NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects

NameHash [main]
NameHash.pro
Sources
NameHash.cpp

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash()
23 * in main and simpio.h
24 */
25 int nameHash(string s);
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 function that computes the hash.
38 * to talk more about what hash functions do later
39 * the meantime, think of it as a function that scans
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined,
43 * treats each character in the input name as a number.
```

This is pretty cool, actually!

Name	Value	Type
first	"Ada"	std::string

returned value 1967457 int

Name	Value	Type
------	-------	------

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "function-finished".

Level	Function	File	Line	Address	Number	Funct File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x5555555b6595							
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x55555556161bc							
3	GThreadStd::run()			0x5555555f9476							
4	??			0x7ffff6143d84							
5	start_thread	pthread_create.c	463	0x7ffff6257590							
6	clone	clone.S	95	0x7ffff5e30223							

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results



Activities Qt Creator Jan 4 4:02 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash
23 * in main and
24 */
25 int nameHash(string first, string last);
26
27 int main() {
28     string first;
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 function that computes the hash.
38 * to talk more about what hash functions do later
39 * the meantime, think of it as a function that scans
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined,
43 * treats each character in the input name as a number.
```

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!

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "function-finished".

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x555555b6595								
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
3	GThreadStd::run()			0x5555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

returned value 1967457 int

Name	Value	Type
------	-------	------



Jan 4 4:02 PM

NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects

NameHash [main]
NameHash.pro
Sources
NameHash.cpp

```
19 #include "simpio.h"
20 using namespace std;
21
22 /* Prototype for nameHash()
23 * in main and simpio.h
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 function that computes the hash value.
38 * to talk more about what hash functions do later
39 * the meantime, think of it as a function that scans
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inclined,
43 * treats each character in the input name as a number.
```

Let's do a "step over" so that we can finish executing this line. Click "step over," and if you did everything right...

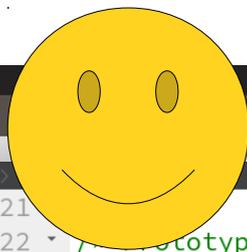
Name	Value	Type
first	"Ada"	std::string
last	"Lovelace"	std::string
returned value	1967457	int



Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "function-finished".

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	31	0x5555555b6595								
2	std::_Function_handler<int (), QtGui::startBackgroundEventLoop...>::operator()			0x55555556161bc								
3	GThreadStd::run()			0x55555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								

Type to locate (Ctrl...) 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results



Activities Qt Creator Jan 4 4:07 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects

- NameHash [main]
- NameHash.pro
- Sources
- NameHash.cpp

```
21
22 /* Prototype for the nameHash function. This lets u
23 * in main and then de
24 */
25 int nameHash(s
26
27 int main() {
28     string fir
29     string las
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashVal
34     return 0;
35 }
36
37 /* This is the actual function that computes the ha
38 * to talk more about what hash functions do later
39 * the meantime, think of it as a function that scr
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inc
43 * treats each character in the input name as a num
44 * It then uses them as coefficients in a polynomia
45 * F_p, where p is a large prime number, and evalu
```

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int
last	"Lovelace"	std::string

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

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Func	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	33	0x555555b65b3								
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
3	GThreadStd::run()			0x5555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results



At this point, we've seen just about everything we care about. Rather than single-stepping all the way to the end, let's just tell the program to keep on running.

Activities Qt Creator Jan 4 4:07 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
21
22 /* Prototype for
23 * in main and
24 */
25 int nameHash(s
26
27 int main() {
28     string fir
29     string last = get
30
31     int hashValue = nameHash(first, last);
32
33     cout << "The hash of your name is: " << hashVal
34     return 0;
35 }
36
37 /* This is the actual function that computes the ha
38 * to talk more about what hash functions do later
39 * the meantime, think of it as a function that scr
40 * of the input and produces a number.
41 *
42 * For those of you who are more mathematically inc
43 * treats each character in the input name as a num
44 * It then uses them as coefficients in a polynomi
45 * F_p, where p is a large prime number, and evalu
```

Name	Value	Type
first	"Ada"	std::string

Debugger GDB for "NameHash" Threads: #12 NameHash Stopped: "end-stepping-range".

Level	Function	File	Line	Address	Number	Funct	File	Line	Address	Condition	Ignore	Threads
1	studentMain	NameHash.cpp	33	0x555555b65b3								
2	std::_Function_handler<int (), QtGui::startBackgroundEve...			0x5555556161bc								
3	GThreadStd::run()			0x5555555f9476								
4	??			0x7ffff6143d84								
5	start_thread	pthread_create.c	463	0x7ffff6257590								
6	clone	clone.S	95	0x7ffff5e30223								

Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results

```

21
22  /* Prototype for the nameHash function. This lets u
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: " << hashVal
34      return 0;
35  }
36
37  /* This is the actual function that computes the h
38  * to talk more about what hash functions do later
39  * the meantime, think of it as a function that scr
40  * of the input and prod

```

Name	Value	Type
first	"Ada"	std::string
hashValue	1967457	int
last	"Lovelace"	std::string



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."

Func	File	Line	Address	Condition	Ignore	Threads
3	pthread_std...	??	0x555555519476			
4	??	??	0x7ffff6143d84			
5	start_thread	pthread_create.c 463	0x7ffff6257590			
6	clone	clone.S 95	0x7ffff5e30223			

Activities NameHash Jan 4 4:08 PM NameHash.cpp @ NameHash [main] - Qt Creator

File Edit View Build Debug Analyze Tools Window Help

Projects NameHash [main] NameHash.pro Sources NameHash.cpp

```
21
22 /* Prototype for the nameHash function. This lets u
23 * in main and then define it later in the program.
24 */
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

NameHash Console [Completed]

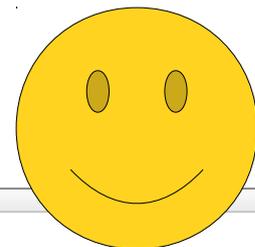
File Edit Options Help

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

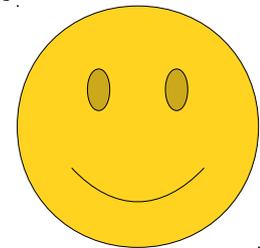
If you do, you should see something like this.
(The program window might not automatically pop up. That's okay! Just open it manually.)
Our program is now done running!

Level	File	Line	Address	Condition	Ignore	Threads
1	student					
2	std::Function_handler<int ()		0x5555556181b0			
3	GThreadStd::run()		0x5555555f9476			
4	??		0x7ffff6143d84			
5	start_thread	pthread_create.c 463	0x7ffff6257590			
6	clone	clone.S 95	0x7ffff5e30223			

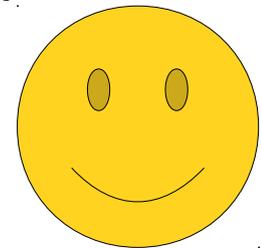
Type to locate (Ctrl... 1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 QML Debugger Console 7 Version Control 8 Test Results



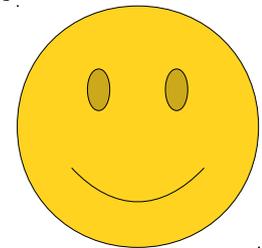
So there you have it! You've now gotten more familiar with the debugger!



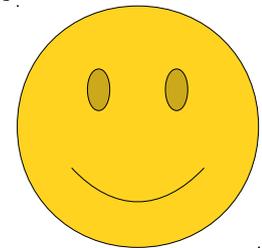
You know how to set a breakpoint to pause the program at a particular point.



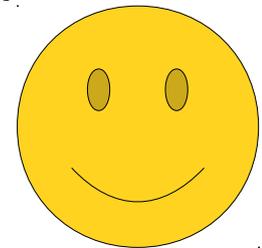
You know how to read the call stack and to see the values of local variables.



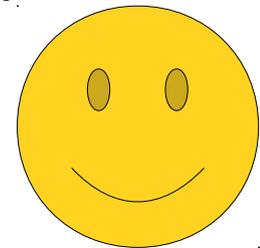
You know how to single-step the program and see what values change.



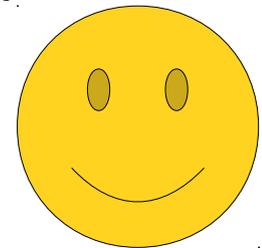
You know how to run a function to completion,
and how to let the program keep on running.



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!

