C++ How can I set the parameter of a `std::exponential distribution` object?

Ask Question

Asked 4 years, 3 months ago Active 4 years, 3 months ago Viewed 489 times



C++ Standard Library - Random Number Generation and Distributions: How can I set the parameter of the exponential distribution?



I have a program which requires exponentially distributed random numbers. I am using the C++11 Random Numbers and Distributions Library support.



I have a distribution: std::exponential_distribution<double> exp_dis(lambda);



lambda is any value to begin with. 0.0 or 1.0 are okay values to use.

I refer to this distribution in a thread function using a pointer. (I have an independent distribution for each thread function to avoid data race conditions.)

The value of lambda is computed within a loop, and can change each time the loop is executed.

Therefore I would like to know how the value of the parameter lamda can be "set" within the exponential distribution.

From some quick searches, I think I should be able to do this using the member function param(), but I can't figure out the exact syntax to use.

This doesn't work:



2 Answers

active oldest votes



You can also change the lambda for your exponential distribution as following.







template<typename T> void set_new_lambda(std::exponential_distribution<T> *exp_dis, T val) typename std::exponential_distribution<T>::param_type new_lambda(val); exp_dis->param(new_lambda);

And can use it as following

```
int main()
  std::exponential_distribution<double> exp_dis(0.1);
  std::cout<<exp_dis.lambda()<<'\n';</pre>
  set_new_lambda(&exp_dis, 0.2);
  std::cout<<exp_dis.lambda()<<'\n';</pre>
```

```
return 0;
}
```

Or may be if you are dealing with only double type then, you can do as following as well.

```
int main()
{
   std::exponential_distribution<double> exp_dis(0.1);
   auto ptr = &exp_dis;
   std::exponential_distribution<double>::param_type new_lambda(0.2);
   ptr->param(new_lambda);
}
```

And you can see that param_type for distribution can be declared as

std::exponential_distribution<double>::param_type .

share improve this answer

edited Oct 27 '15 at 10:55

answered Oct 27 '15 at 4:29



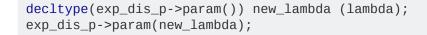
jha-G ■ **1,768** • 10 • 27

add a comment



With the following you should be able to set the new lambda:







This is code I've been using since some time. Like explained by Praetorian in the comments, the param() type has the same arguments as the parent type.

I found this about it in a document about the C++ standardizing:

For each of the constructors of D taking arguments corresponding to parameters of the distribution, P shall have a corresponding constructor subject to the same requirements and taking arguments identical in number, type, and

default values. Moreover, for each of the member functions of D that return values corresponding to parameters of the distribution, P shall have a corresponding member function with the identical name, type, and semantics.

With D being the distribution class, and P is the type named by D's associated param type.

The <u>decltype</u> function simply:

Inspects the declared type of an entity or queries the type and value category of an expression.

share improve this answer

edited Oct 26 '15 at 16:25

answered Oct 26 '15 at 16:03



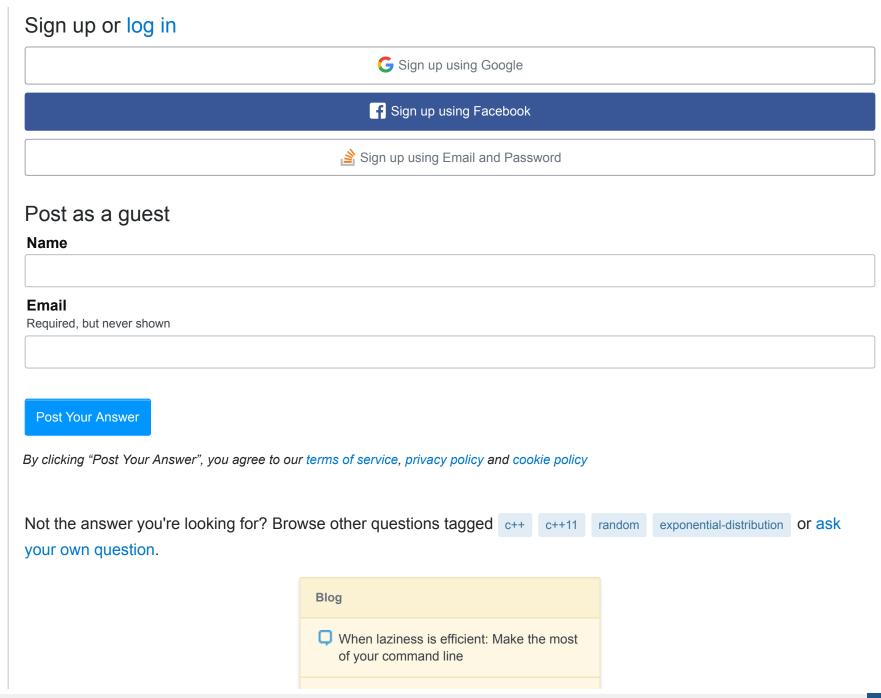
5.048 • 7 • 33 • 49

@user3728501 For any distribution type, the nested param_type type is guaranteed to be constructible using the same constructor arguments as the parent type. – Praetorian Oct 26 '15 at 16:10

add a comment

Your Answer

Ø 66 {} ▲ E E E



□ Ben Popper is the worst coder in the world: Something awry with my array
 Featured on Meta
 □ TLS 1.0 and TLS 1.1 removal for Stack Exchange services
 ☑ Did Stack Exchange cut the number of negative comments nearly in half between...
 ☑ An account of my meeting with the Stack Overflow management team

Related

- How do you set, clear, and toggle a single bit?
- 545 C++: "std::endl" vs "\n"
- How to convert a std::string to const char* or char*?
- How can I profile C++ code running on Linux?
- How can I generate random alphanumeric strings?
- 1848 C++11 introduced a standardized memory model. What does it mean? And how is it going to affect C++ programming?
- Are the days of passing const std::string & as a parameter over?

- Should I call reset() on my C++ std random distribution to clear hidden state?
- Generate random numbers using C++11 random library

Hot Network Questions

- Has there ever been manned mission plans by any Space Agency to go to a solar system object other than the Moon and Mars?
- Why are there waves in seas towards shore even in night?
- Did Lucille Ball have a creative influence on Star Trek?
- Is it unethical for a professor to misrepresent the format of an exam?
- How could Larry get his friends on a plane without knowing their weights?
- Why are no recent small aircraft designed to be "characteristically incapable of spinning" as the Ercoupe was?
- Can Harry Potter swim?
- Corner modelling technique
- Is there any dangerous command in LaTeX?
- Mhat does the horizontal bar indicate in the DMM display?
- Representation Player has trouble imagining the ingame situation
- Can these climates/biomes exist near each other?
- PhD manuscript with grammatical errors and informal phrases
- Reversed lota's

- Is there a hot key that automatically makes a group of vertices into a circle?
- Why is China interested in Socotra Rock?
- Does Iran's sanction play a pivotal role in rejecting papers from a journal?
- Resigning a couple of months before an event : is it abusive resignation?
- Is this a triangle?
- Can you express your answer to this riddle?
- Simplifying constructing native SQL query with nested if-else's
- Series of interestingly increasing words!
- Converting JPG images to PNG
- How to convert graphical images to ASCII art?
- **Question feed**

STACK OVERFLOW	PRODUCTS	COMPANY	STACK EXCHANGE NETWORK	Blog Facebook Twitter LinkedIn
Questions	Teams	About		
			Technology 🖸	
Jobs	Talent	Press	Life / Arts 🕑	
Developer Jobs Directory	Advertising	Work Here	Culture / Recreation 🕟	
Salary Calculator	Enterprise	Legal	Science >	
Help		Privacy Policy	Other >	
Mobile		Contact Us		site design / logo © 2020 Stack Exchange Inc; user
Disable Responsiveness				contributions licensed under cc by-sa 4.0 with attribution required. rev 2020.2.13.36069