Time	Narration
00:01	C اور C ++ میں Increment اور Decrement آپریٹرز پر اسپوکن ٹیوٹوریل میں خوش آمدید.
00:08	اس ٹیوٹوریل میں، ہم نے مندرجہ ذیل کے بارے میں سیکھیں گے:
00:10	Increment اور decrement آپریٹرز
00:12	+ + مثلاً، a + + جو postfix increment (پوسٹفکس انکریمینٹ) آپریٹر ہے.
00:18	a + + - prefix increment (پرِفِکس انکریمینٹ) آپریٹر ہے.
00:22	- - جیسے، a- - ایک postfix decrement (پوسٹ فِکس ڈکریمینٹ) آپریٹر ہے.
00:27	a - - ایک prefix decrement (پرِفِکس ڈکریمینٹ) آپریٹر ہے.
00:31	ہم Type casting کے بارے میں بھی سیکھیں گے.
00:35	اس ٹیوٹوریل کو ریکارڈ کرنے کے لئے، میں- آپریٹنگ سسٹم کے طور پر استعمال کر رہا ہوں Ubuntu 11.10
00:40	Ubuntu میں gcc اور g + + کمپائلر ورژن 4.6.1
00:48	+ + آپریٹر operand کی موجودہ ویلیو میں ایک کا اضافہ کرتا ہے.
00:54	a + + اور a + +، a = a + 1 کے برابر ہیں.
01:00	-- آپریٹر operand کے موجودہ ویلیو میں ایک کی کمی کرتا ہے.
01:06	--a اور a--، a = a - 1 کے برابر ہیں.
01:13	میں ایک C پروگرام کی مدد سے increment اور decrement آپریٹرز کے استعمال کو سمجھاوں گا.
01:19	میں نے پہلے ہی پروگرام بنالیا ہے، تو میں کوڈ سمجھاوں گا.
01:25	یہاں، ہمارے پاس C میں increment اور decrement آپریٹرز کے لئے کوڈ ہے.
01:30	یہاں، میں نے ایک اِنٹجر ویریبل a لیا ہے، جس کی ویلیو 1 ہے.
01:35	اس طرح ہم a کی ویلیو میں تبدیلیوں کا جائزہ لینے کے قابل ہوں گے.
01:39	اس سے ہم آپریٹرز کے طریقہ کار کے بارے میں بہتر جانیں گے.
01:47	دیکھتے ہیں، کہ postfix increment آپریٹر کس طرح کام کرتا ہے.
01:51	printf اسٹیٹمینٹ کا آوٹ پُٹ 1 ہے.
01:55	ویلیو نہیں بدلے گی.

01:57	اس کی وجہ یہ ہے کیونکہ postfix آپریشن، operand کی جانچ کے بعد لاگو ہوتا ہے.
02:04	اگر + + a پر کوئی آپریشن لاگو ہوتا ہے، تو یہ a کی موجودہ ویلیو پر کیا جاتا ہے.
02:10	اس کے بعد a کی ویلیو میں اضافہ کیا جاتا ہے.
02:17	اب اگر ہم یہاں a کی ویلیو دیکھتے ہیں، تو اس میں 1 کا اضافہ ہوا ہے.
02:27	ہم دوبارہ تبدیلیوں کو ظاہر کرنے کے لئے a کو 1 میں انشلائز کرتے ہیں.
02:35	اب ہم prefix increment آپریٹرز پر آتے ہیں.
02:38	یہ printf اسٹیٹمینٹ سکرین پر 2 پرنٹ کرتا ہے.
02:42	اس کی وجہ یہ ہے کیونکہ prefix آپریشن operand کی جانچ سے پہلے لاگو ہوتا ہے.
02:49	لہذا پہلے a کی ویلیو میں 1 کا اضافہ ہوتا ہے اور پھر اسے پرنٹ کیا جاتا ہے.
02:58	ہم دوبارہ اس کو دیکھنے کے لئے a کی ویلیو کو پرنٹ کرتے ہیں کہ اس میں کوئی اور تبدیلی نہیں تو نہیں ہوئی.
03:03	اب اس کوڈ کو ایکزِکیوٹ کر کے چیک کرتے ہیں.
03:07	میں مندرجہ ذیل لائنز کمینٹ کروں گا . ٹائپ کریں /* ، */
03:19	Save پر کلک کریں.
03:22	میں نے اپنی فائل incrdecr.c نام سے سیو ہے.
03:29	اپنے کی بورڈ پر Ctrl، Alt اور' T کیز 'ایک ساتھ دبا کر ٹرمنل ونڈو کھولیں.
03:35	کمپائل کرنے کے لئے، ٹرمنل پر مندرجہ ذیل ٹائپ کریں gcc space incrdecr dot c space minus o space incr. انٹر دبائیں.
03:51	کوڈ ایکزِکیوٹ کرنے کے لئے، './incr' ٹائپ کریں . انٹر دبائیں.
03:59	آؤٹ پٹ سکرین پر دکھایا جاتا ہے.
04:01	یہ آؤٹ پٹ ہے، جب آپ + + a پرنٹ کرتے ہیں.
04:06	یہ آؤٹ پٹ ہے، جب آپ a + + پرنٹ کرتے ہیں.
04:09	ہم دیکھ سکتے ہیں، کہ نتیجہ پہلے بتائے گئے طریقے کے جیسا ہی ہے.
04:13	اب باقی پروگرام پر واپس آئیں.
04:16	میں اب postfix اور prefix decrement آپریٹرز کے بارے میں بتاؤں گا.

04:21	یہاں اور یہاں سے ملٹی لائن حذف کریں.
04:29	اب ہم دوبارہ a کو 1 ویلیو منسوب کرتے ہیں.
04:35	یہ printf اسٹیٹمینٹ آؤٹ پُٹ 1 ویلیو دیتا ہے، جیسا کہ پہلے بیان کیا گیا ہے.
04:40	A کی ویلیو میں a-- کی تشخیص کے بعد 1 کی کمی ہوگی، چونکہ یہ ایک postfix expression ہے.
04:47	اگلا اسٹیٹمینٹ a کی ویلیو 0 پرنٹ کرتا ہے.
04:51	A کی ویلیو، اب 1 سے گھٹ گئی ہے.
04:54	ہمارے پاس اب prefix decrement آپریٹر ہے.
04:58	اس printf اسٹیٹمینٹ کا آؤٹ پُٹ 0 ہے
05:00	کیونکہ یہ ایک prefix آپریشن ہے.
05:05	prefix آپریشن operand کی جانچ کے بعد لاگو ہوتا ہے.
05:09	اس printf اسٹیٹمینٹ کا آؤٹ پُٹ 0 ہے.
05:11	a کی ویلیو میں کوئی دیگر تبدیلی نہیں ہوئی ہے.
05:15	return 0 ٹائپ کریں .اور کلوز کرلی بریکیٹ بند کریں.
05:21	Save پر کلک کریں.
05:24	ٹرمینل پر واپس آئیں.
05:27	کمپائل کرنے کے لئے، ٹرمینل پر مندرجہ ذیل ٹائپ کریں gcc space incrdecr dot c space . minus o space incr، اینٹر دبائیں.
05:42	ایکزِکیوٹ کرنے کے لئے ٹائپ کریں، ./incr، اینٹر دبائیں.
05:52	یہ آؤٹ پُٹ ہے جب آپ a-- پرنٹ کرتے ہیں.
05:56	یہ آؤٹ پُٹ ہے جب آپ --a پرنٹ کرتے ہیں.
05:59	لہذا، اب ہم دیکھتے ہیں، کہ increment اور decrement آپریٹرز کس طرح کام کرتے ہیں.
06:05	اگر ہم C ++ میں یہی پروگرام لکھنا چاہتے ہیں،
06:07	میں مندرجہ بالا C کوڈ میں کچھ تبدیلیاں کرسکتا ہوں.
06:10	ایڈیٹر پر واپس جائیں.

06:13	یہاں ضروری کوڈ پر مشتمل C ++ فائل ہے.
06:16	نوٹ کریں، کہ یہ header، C فائل کے ہیڈر سے مختلف ہے.
06:20	ہمارے پاس using namespace اسٹیٹمینٹ بھی ہے.
06:24	اس کے علاوہ، نوٹ کریں، کہ C ++ میں آؤٹ پُٹ اسٹیٹمینٹ cout ہے.
06:28	تو ان اختلافات کے علاوہ، دونوں کوڈ کافی ملتے جلتے ہیں.
06:33	فائل سیو کریں. فائل cpp. ایکسٹینشن کے ساتھ سیو ہوتی ہے.
06:40	کوڈ کمپائل کریں.
06:42	ٹرمنل کھولیں اور ٹائپ کریں g + + space incrdecr dot cpp space minus o space incr اینٹر دبائیں.
07:00	ایکزِیکیوٹ کرنے کے لئے ٹائپ کریں incr /. اینٹر دبائیں.
07:07	آؤٹ پٹ سکرین پر دکھایا جاتا ہے.
07:10	ہم دیکھتے ہیں، کہ آؤٹ پٹ C پروگرام کی طرح ہی ہے.
07:15	اب ہم typecasting کے تصور کو سمجھتے ہیں.
07:17	یہ C اور C ++ دونوں میں ایک جیسے طریقے سے لاگو ہوتا ہے.
07:22	Typecasting کا استعمال ایک قسم کے ویریبل کو دوسرے قسم کے ویریبل کی طرح کام کرنے کے قابل بنانے کے لئے کیا جاتا ہے.
07:27	Typecasting آپ کے مطلوبہ ڈیٹا ٹائپ کو بریکٹ میں بند کر کے کیا جاتا ہے.
07:33	اس cast کو ویریبلس کے سامنے رکھا جاتا ہے، جسے آپ cast کرنا چاہتے ہیں.
07:38	یہ typecast صرف ایک واحد آپریشن کے لئے کارگر ہوتا ہے.
07:42	اب a ایک آپریشن کے لئے ایک float ویریبل کی طرح کام کرے گا.
07:47	یہاں ایک مثال ہے، جو میں نے پہلے ہی بنا لی ہے.
07:50	میں اب کوڈ سمجھاوں گا.
07:54	ہم پہلے اِنٹجر کے طور پر ویریبل a اور b، اور فلوٹ کے طور پر c ڈکلیئر کریں گے.
08:00	a کو 5 ویلیو دی ہے۔ b کو 2 ویلیو دی ہے.

08:06	ہم a اور b پر آپریشنس کریں گے.
08:10	ہم a کو b سے تقسیم کرتے ہیں . تقسیم کا نتیجہ c میں اسٹور کیا جاتا ہے.
08:14	ہم نے عشاریہ کے دو مقامات تک درستگی کو ظاہر کرنے کے لئے 2f.% استعمال کیا ہے.
08:20	ظاہر کیا گیا نتیجہ متوقع نتیجے 2.50 کے بجائے 2.00 ہے
08:25	عشاریائی حصہ ہٹا دیا جاتا ہے، کیونکہ دونوں ہی آپرینڈس a اور b اِنٹجر ہیں.
08:31	real division کرنے کے لئے، ایک operand میں فلوٹ کے لئے type cast ہونا چاہئے
08:35	یہاں ہم a کو فلوٹ میں typecast کر رہے ہیں۔ c میں اب real division کی ویلیو اسٹور ہے.
08:41	اب real division کا نتیجہ ظاہر کیا جاتا ہے . جواب 2.50 ہے جو حسبِ توقع ہے.
08:47	return 0؛ ٹائپ کریں اور کلوز کرلی بریکٹ بند کریں.
08:51	Save پر کلک کریں .فائل c. ایکسٹینشن کے ساتھ سیو کریں.
08:55	میں نے اپنی فائل typecast.c نام سے سیو کی ہے.
08:59	ٹرمینل کھولیں.
09:01	کمپائل کرنے کے لئے، ٹائپ کریں gcc space typecast dot c space minus o space type اینٹر دبائیں.
09:17	ایکزِکیوٹ کرنے کے لئے، ٹائپ کریں type/. اینٹر دبائیں.
09:25	آؤٹ پٹ سکرین پر دکھایا جاتا ہے.
09:27	دونوں ویلیوس کو دیکھتے ہوئے، ہم typecasting کے اثرات کو سمجھ سکتے ہیں.
09:32	اب ہم ٹیوٹوریَل کا خلاصہ کریں گے.
09:34	اس ٹیوٹوریل میں، ہم نے سیکھا کہ،
09:36	increment اور decrement آپریٹرز کا استعمال کیسے کریں.
09:40	ہم نے Prefix اور Postfix فارمیٹس کے بارے میں بھی سیکھا.
09:44	اس کے علاوہ ہم نے typecasting اور اس کے استعمال کے طریقوں کے بارے میں بھی سیکھا.
09:47	ایک مشق کے طور پر:

09:49	مندرجہ ذیل expression کو حل کرنے کے لئے، ایک پروگرام لکھیں، a divided by b plus c divided by d.
09:56	a، b، cاور d کی ویلیوان پٹ کے طور پر یوزر سے حاصل ہوگی.
10:01	real division کے لئے typecasting کا استعمال کریں.
10:05	مندرجہ ذیل لنک پر دستیاب ویڈیوز دیکھیں..
10:08	یہ اسپوکن ٹیوٹوریل پروجیکٹ کا خلاصہ کرتا ہے.
10:10	اگر آپ کے پاس اچھی بینڈ وڈتھ نہیں ہے تو آپ اسے ڈاؤن لوڈ کر کے بھی دیکھ سکتے ہیں.
10:15	اسپوکن ٹیوٹوریل پروجیکٹ ٹیم،
10:17	اسپوکن ٹیوٹوریلس کا استعمال کرتے ہوئے ورکشاپس منعقد کرتی ہے.
10:20	اور جو آن لائن ٹیسٹ پاس کرتے ہیں انہیں سند بھی دیتے ہیں.
10:24	مزید معلومات کے لئے contact at spoken hyphen tutorial dot org پر لکھیں.
10:33	اسپوکن ٹیوٹوریل پروجیکٹ، ٹاک ٹو اے ٹیچر پروجیکٹ کا حصہ ہے.
10:37	یہ بھارت حکومت کے ایمیچارڈی کے "آئی سی ٹی کے ذریعے قومی خواندگی مشن کی طرف سے حمایت شدہ ہے.
10:44	اس مشن پر مزید معلومات spoken hyphen tutorial dot org slash NMEICT hyphen Intro پر دستیاب ہے.
10:55	اس اسکرپٹ کا ترجمہ اور صدا بندی میں نے یعنی وجاحت احمد نے کی ہے، شامل ہونے کیلئے آپکا شکریہ