randint operator

Files
randint-inl.h
randint.cc
randint.cu

randint-inl.h

```
#ifndef MXNET_OPERATOR_RANDOM_RANDINT_INL_H_
#define MXNET_OPERATOR_RANDOM_RANDINT_INL_H_
```

Because - src/operator/random/randint-inl.h

Included all the other base files just like other random files

Quadratic-op-inl.h

1. Define Param
2. OpShape
3. OpType
4. OpStorageType

```
struct QuadraticParam : public dmlc::Parameter<QuadraticParam>

inline bool QuadraticOpShape(const nnvm::NodeAttrs& attrs,
                             std::vector<int>* in_attrs,
                             std::vector<int>* out_attrs

inline bool QuadraticOpType(const nnvm::NodeAttrs& attrs,
                            std::vector<int>* in_attrs,
                            std::vector<int>* out_attrs

inline bool QuadraticOpStorageType(const nnvm::NodeAttrs& attrs,
                                   const int dev_mask,
                                   DispatchMode* dispatch_mode,
                                   std::vector<int>* in_attrs,
                                   std::vector<int>* out_attrs)

template<int req>
struct quadratic_forward

template<int req>
struct quadratic_backward

template<typename xpu>
void QuadraticOpForward(const nnvm::NodeAttrs& attrs,
                        const OpContext& ctx,
                        const std::vector<TBlob>& inputs,
                        const std::vector<OpReqType>& req,
                        const std::vector<TBlob>& outputs)
```

randint.cc

Imports base file (header file) i.e. randint-inl.h

Point of diff

1. Backend op registration quip doc says - NNVM
    a. Quadratic op (Beginer guide) - DMLC_DECLARE_PARAMETER

Order of execution

1. test_random.py : `mx.nd.random.randint`
2. python/mxnet/ndarray/random.py : `randint()` → `_random_helper()` → `random()`
3. python/mxnet/_ctypes/ndarray.py: `_imperative_invoke()`

stype? ctype?
?? goes somewhere??

1. src/operator/random/sample_op.h: `SampleOpType()`
2. src/operator/tensor/init_op.h: `InitStorageType()`
3. src/imperative/imperative.cc : `Imperative::Invoke()`
4. src/imperative/imperative_utils.h: `SetShapeType()` → `FInferShape, FInferType` and `FInferStorageType`

`struct SampleMaster<RandIntSampler>{}`

MXImperativeInvokeEx → MXImperativeInvokeImpl →

RNG - random number generator

- mt19937 is a standard mersenne_twister_engine

Issues

- static assertion failed template argument not an integral type
    - Understood following things
    - typedef typename A
    - `std::is_floating_point<A>::value` evaluates to False/True
        - Eg - `std::is_floating_point<float>::value` → True

cuRand - Cuda Random
Types of RNG

- Philox
- MT19937 - Mersenne Twister
- dSFMT - Double SIMD Mersenne Twister
- PCG64 - Parallel Congruent Generator (64-bit, PCG64)
- ThreeFry Counter-based RNG

```
// typedef typename std::conditional<std::is_integral<DType>::value,
      // std::uniform_int_distribution<DType>,
      // std::uniform_real_distribution<FType>>::type GType;
      // GType dist_discrete_uniform(lower, upper);
```