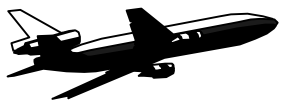
SIMULATION



Simulation

- the use of one system to imitate the behavior of another system
  - physical simulations
  - mathematical simulations
  - computer simulations

Computer Simulation

- Objects being studied are represented by data.
- Actions in the system being studied are represented by operations on the data.
- Rules describing these actions are translated into algorithms
- Queues are important data structures in computer simulation
  - objects and actions are usually kept in queues

Simulation of an Airport

- A small airport with one runway.
- One plane can either land or take off in each unit of time.
- Allow a plane to take off only if there are no planes waiting to land.
- Planes arrive ready to land or take off at random times.
- Need two queues, `qLanding` and `qTakeoff`.

Simulation of an Airport

```
void main( void )
{
  QueueType     qLanding, qTakeoff;
  PlaneType     plane;
  int           curtime, n, i;

  for( curtime = 1; curtime <= 100; curtime++ ) {
    n = RandomNumber();
    for ( i=1; i<=n; i++ ) {
      NewPlane( &plane, ARRIVE );
      QueueIsFull( &qLanding ) ? Refuse( plane )
                               : AddQueue( plane, &qLanding);
    }
    n = RandomNumber();
    for ( i=1; i<=n; i++ ) {
      NewPlane( &plane, DEPART );
      QueueIsFull( &qTakeoff ) ? Refuse( plane )
                               : AddQueue( plane, &qTakeoff);
    }
  ...
```

Simulation of an Airport

```
void main( void )
{
    ...

    if ( ! QueueIsEmpty( &qLanding ) ) {
      DeleteQueue( &plane, &qLanding );
      Land( plane, curtime );
    } else
    if ( ! QueueIsEmpty( &qTakeoff ) ) {
      DeleteQueue( &plane, &qTakeoff );
      Takeoff( plane, curtime );
    } else
      Idle( curtime );
  }
  Conclude( &qLanding, &qTakeoff );
}
```

Simulation of an Airport

```
typedef  enum  actionTag {ARRIVE, DEPART} ActionType;
typedef  struct planeTag {
         int    id;               /* id number of airplane    */
         int    time;         /* time of arrival in queue */
} PlaneType;

void main( void )
{
  QueueType    qLanding, qTakeoff;
  PlaneType    plane;
  int              curTime, endTime;
  int              idleTime, landWait, takeoffWait;
  int              nLand, nTakeoff, nRefuse;
  int              nPlanes;
  int              i, n;
  double           expectArrive, expectDepart;

  InitializeQueue( &qLanding ); InitializeQueue( &qTakeoff );
  idleTime = landWait = takeoffWait = 0;
  nLand = nTakeoff = nRefuse = nPlanes = 0;
  Start( &endTime, &expectArrive, &expectDepart );
```

Simulation of an Airport

```
for ( curTime = 1; curTime <= endTime; curTime++ ) {
  n = RandomNumber( expectArrive );
  for ( i=1; i<=n; i++ ) {
    NewPlane( &plane, &nPlanes, curTime, ARRIVE );
    QueueIsFull( &qLanding ) ? Refuse( plane, &nRefuse, ARRIVE )
                             : AddQueue( plane, &qLanding );
  }
  n = RandomNumber( expectDepart );
  for ( i=1; i<=n; i++ );
    NewPlane( &plane, &nPlanes, curTime, DEPART );
    QueueIsFull( &qTakeoff ) ? Refuse( plane, &nRefuse, DEPART )
                             : AddQueue* plane, &qTakeoff );
  }
  if ( ! QueueIsEmpty( &qLanding ) {
    DeleteQueue( &plane, &qLanding );
    Land( plane, curTime, &nLand, &landWait );
  } else
  if ( ! QueueIsEmpty( &qTakeoff ) {
    DeleteQueue( &plane, &qTakeoff );
    Takeoff( plane, curTime, &nTakeoff, &takeoffWait );
  } else
    Idle( curTime, &idleTime );
}
```

Simulation of an Airport

```
void NewPlane( PlaneType *pPlane, int *pnPlanes,
               int curTime, ActionType  action )
{
  (*pnPlanes)++;  pPlane->id = *pnPlanes;  pPlane->time = curTime;
  switch( action ) {
    case ARRIVE :
      printf("Plane %3d ready to land\n", *pnPlanes );
      break;
    case DEPART :
      printf("Plane %3d ready to take off\n", *pnPlanes );
      break;
  }
}
void Refuse( PlaneType plane, int *pnRefuse, ActionType action )
{
  (*pnRefuse)++;
  switch( action ) {
    case ARRIVE :
      printf("Plane %3d directed to another airport\n", plane.id );
      break;
    case DEPART :
      printf("Plane %3d told to try later\n", plane.id );
      break;
  }
}
```

Simulation of an Airport

```
void Land( PlaneType plane, int curTime, int *pnLand, int *plandWait )
{
  int         wait;

  wait = curTime - plane.time;
  printf( "%3d : Plane %3d laned; in queue %d units\n",
           curTime, plane.id, wait );
  (*pnLand)++;
  *plandWait += wait;
}
```

```
void Takeoff( PlaneType plane, int curTime,
       int *pnTakeoff, int *ptakeoffWait )
{
  int       wait;

  wait = curTime - plane.time;
  i tf( "%3d  Pl    %3d t  k  ff  i        %d    it \  "
```