

本节目标

- 绘制如下粒子时钟

一、渲染粒子数字

首先来看一下如何渲染出如下的粒子的数字。



1.数字渲染思路

前面我们知道，对于粒子的东西，最重要的就是粒子的属性信息，尤其是位置。前面说过的使用图片资源进行文字的渲染，实际上就是通过颜色像素在图片中的位置来映射出粒子的位置。这里也是类似。如下左图，我们用一个二维数组维护一个数字的显示。其中1时我们就添加粒子。



下面是显示出 1994 粒子的方法。通过 collectParticles 进行收集粒子。

```
collectParticles() {
  collectDigit(target: 1, index: 0);
  collectDigit(target: 9, index: 1);
  collectDigit(target: 9, index: 2);
  collectDigit(target: 4, index: 3);
}

void renderDigit(int target = 0, int index = 0) {
  if (target > 10) {
    return;
  }
  double offsetx = 0;
  double space = _radius;
  offsetx = (digits[target][0].length * 2 * (_radius + 1) + space * 2) * index;
  for (int i = 0; i < digits[target].length; i++) {
    for (int j = 0; j < digits[target][i].length; j++) {
      if (digits[target][i][j] == 1) {
        double rx = j * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        double ry = i * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        particles[count] = Particle(x: offsetx + rx,
                                     y: ry,
                                     size: _radius, color: Colors.blue);
        count++;
      }
    }
  }
}
```

2.粒子信息类和管理器

将粒子的属性信息为此定一个类。

```
class Particle {
  double x; // 粒子坐标
  double y; // 粒子坐标
  double size; // 粒子大小
  Color color; // 粒子颜色
  bool active; // 粒子是否可用
  double vx; // 粒子水平速度
  double vy; // 粒子垂直速度
  double vx; // 粒子垂直速度
  double vy; // 粒子垂直速度
}

Particle({
  this.x = 0,
  this.y = 0,
  this.size = 0,
  this.color = Colors.black,
});
```

粒子管理器中在构造方法里对粒子列表进行初始化。更新时根据索引对粒子信息进行修改即可。而不是频繁的添加或删除。每次 tick 方法被调用时，会根据时间来更新所有粒子信息。这时时钟的效果。

```
class ClockManager with ChangeNotifier {
  List<Particle> particles; // 粒子列表
  int numParticles; // 粒子数量
  Size size; // 尺寸
  DateTime datetime; // 时间
}

ClockManager(this.size, this.numParticles = 500) {
  particles = List<Particle>();
  datetime = DateTime.now();
}

collectParticles() {
  collectDigit(target: 1, index: 0);
  collectDigit(target: 9, index: 1);
  collectDigit(target: 9, index: 2);
  collectDigit(target: 4, index: 3);
}

int count = 0;
double _radius = 6;

void collectDigit(int target = 0, int index = 0) {
  // 同上
}

void tick(DateTime datetime) {
  collectParticles(datetime);
  notifyListeners();
}
```

3.组件类和绘制类

在上一篇文章中使用了 Ticker 类触发时间流，就消除了 AnimationController 对动画的依赖。初始化状态时，实例化 ClockManager 和 Ticker，并让粒子管理器中收集粒子。

```
class ClockPanel extends StatefulWidget {
  @override
  _ClockPanelState createState() => _ClockPanelState();
}

class _ClockPanelState extends State<ClockPanel> with SingleTickerProviderStateMixin {
  Ticker _ticker;
  ClockManager pm;

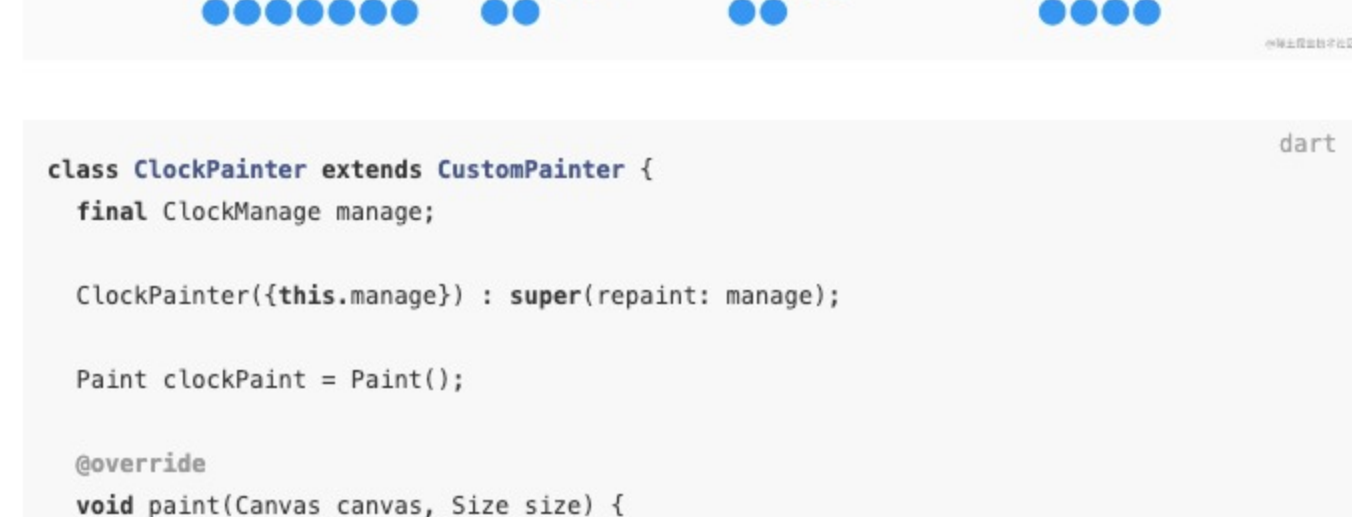
  @override
  void initState() {
    super.initState();
    pm = ClockManager(size: Size(400, 300));
    pm.collectParticles();
    _ticker = createTicker(_tick)..start();
  }

  @override
  void dispose() {
    _ticker.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return CustomPaint(
      size: pm.size,
      painter: ClockPainter(manager: pm),
    );
  }

  void _tick(Duration duration) {
    if (DateTime.now().second % 1 == 0) pm.tick(DateTime.now());
  }
}
```

绘制很简单，遍历画面即可。经过这几个类的协作，粒子的时钟便跃然纸上。



```
class ClockPainter extends CustomPainter {
  final ClockManager manager;

  ClockPainter(this.manager) : super(repaint: manager);

  Paint clockPaint = Paint();

  @override
  void paint(Canvas canvas, Size size) {
    manager.particles.where((e) => e.isActive).forEach((particle) {
      clockPaint.color = particle.color;
      canvas.drawCircle(
        Offset(particle.x, particle.y), particle.size, clockPaint);
    });
  }

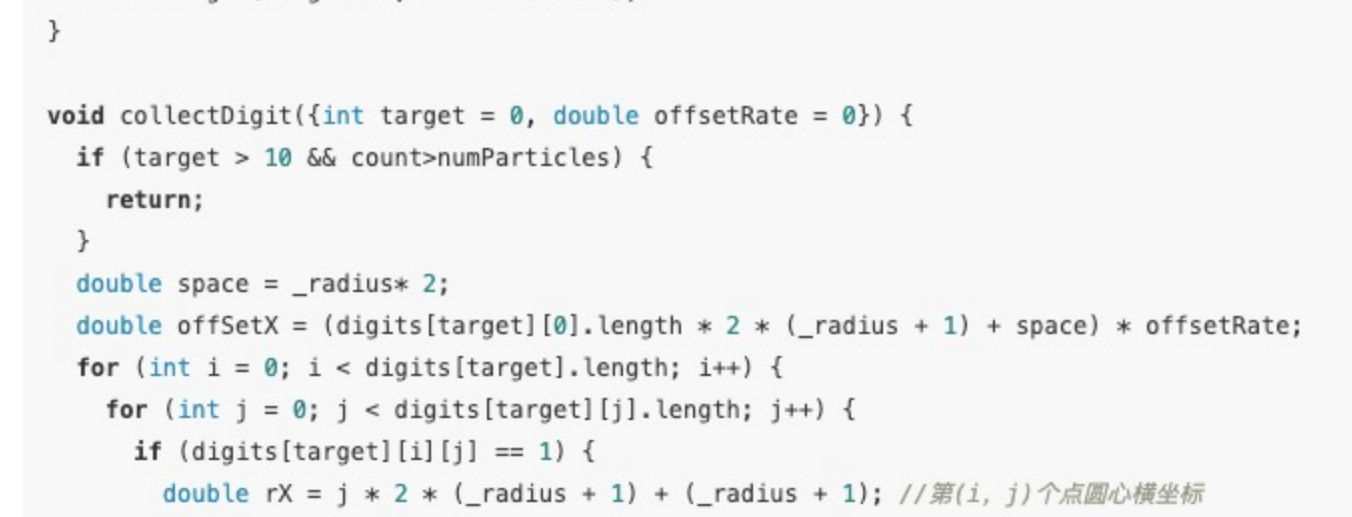
  @override
  bool shouldRepaint(covariant CustomPainter oldDelegate) => false;
}
```

二、粒子时间的绘制

前面虽然实现了粒子数字的绘制，但对于时间来说还是有些细节，比如：会影响粒子的定位。

1.粒子时间静态效果

通过一个自定义的偏移来校正位置。收集粒子后是如下效果。接下来我们只调整时间，重设数字即可



```
collectParticles() {
  collectDigit(target: 0, offsetRate: 0);
  collectDigit(target: 5, offsetRate: 1);
  collectDigit(target: 10, offsetRate: 3.2);
  collectDigit(target: 2, offsetRate: 2.3);
  collectDigit(target: 8, offsetRate: 3.3);
  collectDigit(target: 10, offsetRate: 7.25);
  collectDigit(target: 4, offsetRate: 3);
  collectDigit(target: 9, offsetRate: 6);
}

void collectDigit(int target = 0, double offsetRate = 0) {
  if (target > 10 && count > numParticles) {
    return;
  }
  double space = _radius * 2;
  double offsetx = (digits[target][0].length * 2 * (_radius + 1) + space) * offsetRate;
  for (int i = 0; i < digits[target].length; i++) {
    for (int j = 0; j < digits[target][i].length; j++) {
      if (digits[target][i][j] == 1) {
        double rx = j * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        double ry = i * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        particles[count] = Particle(x: offsetx + rx,
                                     y: ry,
                                     size: _radius, color: Colors.green);
        count++;
      }
    }
  }
}
```

2.时间的数字解析

我们只需要将时间解析成数字即可，也非常简单。有一点要注意的是：我们未来避免对列表的数据插入和删除操作，使用了定长的列表对粒子索引进行修改。这样会存在一个问题，比如1->2时，只是标识了2对应的粒子，可能有些索引是上次在1中的粒子，但不存在2中，无法清除掉，可用在 Particle 中使用一个 active 标识是否控制粒子是否被清除。在 collectParticles 时把所有粒子的 active 标识设为 false，在修改时设置为 true。

```
collectParticles(DateTime datetime) {
  count = 0;
  particles.forEach((Particle element) {
    if (element.isActive) {
      element.active = false;
    }
  });
  collectDigit(target: datetime.hour ~/ 10, offsetRate: 0);
  collectDigit(target: datetime.hour % 10, offsetRate: 1);
  collectDigit(target: 10, offsetRate: 3.2);
  collectDigit(target: datetime.minute ~/ 10, offsetRate: 2.5);
  collectDigit(target: datetime.minute % 10, offsetRate: 3.3);
  collectDigit(target: 10, offsetRate: 7.25);
  collectDigit(target: datetime.second ~/ 10, offsetRate: 5);
  collectDigit(target: datetime.second % 10, offsetRate: 6);
}

void collectDigit(int target = 0, double offsetRate = 0) {
  if (target > 10 && count > numParticles) {
    return;
  }
  double space = _radius * 2;
  double offsetx = (digits[target][0].length * 2 * (_radius + 1) + space) * offsetRate;
  for (int i = 0; i < digits[target].length; i++) {
    for (int j = 0; j < digits[target][i].length; j++) {
      if (digits[target][i][j] == 1) {
        double rx = j * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        double ry = i * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        particles[count] = Particle(x: offsetx + rx,
                                     y: ry,
                                     size: _radius, color: Colors.green,
                                     active: true);
        count++;
      }
    }
  }
}
```

3.时间的运动

为了尽可能少的减少绘制此时，并不需要每次 Ticker 触发时 (16 ms) 都刷新。通过对比毫秒数判断是否大于 1000 即可。这样每秒进行一次粒子时钟的刷新，性能是最好的。通过这些操作，我们就实现了粒子时钟的基本效果。



```
class _ClockPanelState extends State<ClockPanel> with SingleTickerProviderStateMixin {
  Ticker _ticker;
  ClockManager pm;

  @override
  void initState() {
    super.initState();
    pm = ClockManager(size: Size(400, 200));
    _ticker = createTicker(_tick)..start();
  }

  @override
  void dispose() {
    _ticker.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return CustomPaint(
      size: pm.size,
      painter: ClockPainter(manager: pm),
    );
  }

  void _tick(Duration duration) {
    if (DateTime.now().millisecondsSinceEpoch - pm.datetime.millisecondsSinceEpoch > 1000) {
      pm.datetime = DateTime.now();
      _tick();
    }
  }
}
```

三、时钟背景粒子

如下面，在时钟变化时，对应的数字会产生彩色粒子，进行运动。



由于背景粒子 and 粒子时钟的刷新时机不一样，可以使用两个画布。通过 Stack 进行堆叠。由于粒子需要不断进行运动，所以每次 Ticker 时都需要进行刷新。更新在 checkParticles 中检测粒子时变化的数字，进行添加可动的随机粒子。为避免粒子数量爆炸，可以在粒子运动到底端时移除粒子，最终可以将粒子的数量平均控制在 60 左右。这样对于渲染来说压力就不是很大了。

```
class BGManager with ChangeNotifier {
  List<Particle> particles;
  DateTime datetime; // 时间
  Random random = Random();

  // 粒子列表
  int numParticles;

  // 最大粒子数
  Size size; // 尺寸
}

BGManager(this.size, this.numParticles = 500) {
  particles = List<Particle>();
  datetime = DateTime.now();
}

checkParticles(DateTime now) {
  // 判断当前时间是否变化，再决定是否移除粒子
  if (datetime.hour ~/ 10 != now.hour ~/ 10) {
    collectDigit(target: datetime.hour ~/ 10, offsetRate: 0);
  }
  if (datetime.hour % 10 != now.hour % 10) {
    collectDigit(target: datetime.hour % 10, offsetRate: 1);
  }
  if (datetime.minute ~/ 10 != now.minute ~/ 10) {
    collectDigit(target: datetime.minute ~/ 10, offsetRate: 2.5);
  }
  if (datetime.minute % 10 != now.minute % 10) {
    collectDigit(target: datetime.minute % 10, offsetRate: 3.3);
  }
  if (datetime.second ~/ 10 != now.second ~/ 10) {
    collectDigit(target: datetime.second ~/ 10, offsetRate: 5);
  }
  if (datetime.second % 10 != now.second % 10) {
    collectDigit(target: datetime.second % 10, offsetRate: 6);
    datetime = now;
  }
}

double _radius = 4;

void collectDigit(int target = 0, double offsetRate = 0) {
  if (target > 10) {
    return;
  }
  double space = _radius * 2;
  double offsetx = (digits[target][0].length * 2 * (_radius + 1) + space) * offsetRate;
  for (int i = 0; i < digits[target].length; i++) {
    for (int j = 0; j < digits[target][i].length; j++) {
      if (digits[target][i][j] == 1) {
        double rx = j * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        double ry = i * 2 * (_radius + 1) + (_radius + 1); // 画1, 1个点跟圆心重合
        Particle particle = Particle(
          x: rx + offsetx,
          y: ry,
          size: _radius,
          color: randomColor(),
          active: true,
          vx: 2.5 * random.nextDouble() - 1, random.nextInt(200),
          vy: 2 * random.nextDouble() + 1);
        particles.add(particle);
      }
    }
  }
}

Color randomColor() {
  int limitA = 120,
  int limitB = 0,
  int limitC = 0,
  int limitD = 0,
  int limitE = 0,
  int limitF = 0,
  int limitG = 0,
  int limitH = 0,
  int limitI = 0,
  int limitJ = 0,
  int limitK = 0,
  int limitL = 0,
  int limitM = 0,
  int limitN = 0,
  int limitO = 0,
  int limitP = 0,
  int limitQ = 0,
  int limitR = 0,
  int limitS = 0,
  int limitT = 0,
  int limitU = 0,
  int limitV = 0,
  int limitW = 0,
  int limitX = 0,
  int limitY = 0,
  int limitZ = 0,
  int limitAA = 0,
  int limitAB = 0,
  int limitAC = 0,
  int limitAD = 0,
  int limitAE = 0,
  int limitAF = 0,
  int limitAG = 0,
  int limitAH = 0,
  int limitAI = 0,
  int limitAJ = 0,
  int limitAK = 0,
  int limitAL = 0,
  int limitAM = 0,
  int limitAN = 0,
  int limitAO = 0,
  int limitAP = 0,
  int limitAQ = 0,
  int limitAR = 0,
  int limitAS = 0,
  int limitAT = 0,
  int limitAU = 0,
  int limitAV = 0,
  int limitAW = 0,
  int limitAX = 0,
  int limitAY = 0,
  int limitAZ = 0,
  int limitBA = 0,
  int limitBB = 0,
  int limitBC = 0,
  int limitBD = 0,
  int limitBE = 0,
  int limitBF = 0,
  int limitBG = 0,
  int limitBH = 0,
  int limitBI = 0,
  int limitBJ = 0,
  int limitBK = 0,
  int limitBL = 0,
  int limitBM = 0,
  int limitBN = 0,
  int limitBO = 0,
  int limitBP = 0,
  int limitBQ = 0,
  int limitBR = 0,
  int limitBS = 0,
  int limitBT = 0,
  int limitBU = 0,
  int limitBV = 0,
  int limitBW = 0,
  int limitBX = 0,
  int limitBY = 0,
  int limitBZ = 0,
  int limitCA = 0,
  int limitCB = 0,
  int limitCC = 0,
  int limitCD = 0,
  int limitCE = 0,
  int limitCF = 0,
  int limitCG = 0,
  int limitCH = 0,
  int limitCI = 0,
  int limitCJ = 0,
  int limitCK = 0,
  int limitCL = 0,
  int limitCM = 0,
  int limitCN = 0,
  int limitCO = 0,
  int limitCP = 0,
  int limitCQ = 0,
  int limitCR = 0,
  int limitCS = 0,
  int limitCT = 0,
  int limitCU = 0,
  int limitCV = 0,
  int limitCW = 0,
  int limitCX = 0,
  int limitCY = 0,
  int limitCZ = 0,
  int limitDA = 0,
  int limitDB = 0,
  int limitDC = 0,
  int limitDD = 0,
  int limitDE = 0,
  int limitDF = 0,
  int limitDG = 0,
  int limitDH = 0,
  int limitDI = 0,
  int limitDJ = 0,
  int limitDK = 0,
  int limitDL = 0,
  int limitDM = 0,
  int limitDN = 0,
  int limitDO = 0,
  int limitDP = 0,
  int limitDQ = 0,
  int limitDR = 0,
  int limitDS = 0,
  int limitDT = 0,
  int limitDU = 0,
  int limitDV = 0,
  int limitDW = 0,
  int limitDX = 0,
  int limitDY = 0,
  int limitDZ = 0,
  int limitEA = 0,
  int limitEB = 0,
  int limitEC = 0,
  int limitED = 0,
  int limitEE = 0,
  int limitEF = 0,
  int limitEG = 0,
  int limitEH = 0,
  int limitEI = 0,
  int limitEJ = 0,
  int limitEK = 0,
  int limitEL = 0,
  int limitEM = 0,
  int limitEN = 0,
  int limitEO = 0,
  int limitEP = 0,
  int limitEQ = 0,
  int limitER = 0,
  int limitES = 0,
  int limitET = 0,
  int limitEU = 0,
  int limitEV = 0,
  int limitEW = 0,
  int limitEX = 0,
  int limitEY = 0,
  int limitEZ = 0,
  int limitFA = 0,
  int limitFB = 0,
  int limitFC = 0,
  int limitFD = 0,
  int limitFE = 0,
  int limitFF = 0,
  int limitFG = 0,
  int limitFH = 0,
  int limitFI = 0,
  int limitFJ = 0,
  int limitFK = 0,
  int limitFL = 0,
  int limitFM = 0,
  int limitFN = 0,
  int limitFO = 0,
  int limitFP = 0,
  int limitFQ = 0,
  int limitFR = 0,
  int limitFS = 0,
  int limitFT = 0,
  int limitFU = 0,
  int limitFV = 0,
  int limitFW = 0,
  int limitFX = 0,
  int limitFY = 0,
  int limitFZ = 0,
  int limitGA = 0,
  int limitGB = 0,
  int limitGC = 0,
  int limitGD = 0,
  int limitGE = 0,
  int limitGF = 0,
  int limitGG = 0,
  int limitGH = 0,
  int limitGI = 0,
  int limitGJ = 0,
  int limitGK = 0,
  int limitGL = 0,
  int limitGM = 0,
  int limitGN = 0,
  int limitGO = 0,
  int limitGP = 0,
  int limitGQ = 0,
  int limitGR = 0,
  int limitGS = 0,
  int limitGT = 0,
  int limitGU = 0,
  int limitGV = 0,
  int limitGW = 0,
  int limitGX = 0,
  int limitGY = 0,
  int limitGZ = 0,
  int limitHA = 0,
  int limitHB = 0,
  int limitHC = 0,
  int limitHD = 0,
  int limitHE = 0,
  int limitHF = 0,
  int limitHG = 0,
  int limitHH = 0,
  int limitHI = 0,
  int limitHJ = 0,
  int limitHK = 0,
  int limitHL = 0,
  int limitHM = 0,
  int limitHN = 0,
  int limitHO = 0,
  int limitHP = 0,
  int limitHQ = 0,
  int limitHR = 0,
  int limitHS = 0,
  int limitHT = 0,
  int limitHU = 0,
  int limitHV = 0,
  int limitHW = 0,
  int limitHX = 0,
  int limitHY = 0,
  int limitHZ = 0,
  int limitIA = 0,
  int limitIB = 0,
  int limitIC = 0,
  int limitID = 0,
  int limitIE = 0,
  int limitIF = 0,
  int limitIG = 0,
  int limitIH = 0,
  int limitII = 0,
  int limitIJ = 0,
  int limitIK = 0,
  int limitIL = 0,
  int limitIM = 0,
  int limitIN = 0,
  int limitIO = 0,
  int limitIP = 0,
  int limitIQ = 0,
  int limitIR = 0,
  int limitIS = 0,
  int limitIT = 0,
  int limitIU = 0,
  int limitIV = 0,
  int limitIW = 0,
  int limitIX = 0,
  int limitIY = 0,
  int limitIZ = 0,
  int limitJA = 0,
  int limitJB = 0,
  int limitJC = 0,
  int limitJD = 0,
  int limitJE = 0,
  int limitJF = 0,
  int limitJG = 0,
  int limitJH = 0,
  int limitJI = 0,
  int limitJJ = 0,
  int limitJK = 0,
  int limitJL = 0,
  int limitJM = 0,
  int limitJN = 0,
  int limitJO = 0,
  int limitJP = 0,
  int limitJQ = 0,
  int limitJR = 0,
  int limitJS = 0,
  int limitJT = 0,
  int limitJU = 0,
  int limitJV = 0,
  int limitJW = 0,
  int limitJX = 0,
  int limitJY = 0,
  int limitJZ = 0,
  int limitKA = 0,
  int limitKB = 0,
  int limitKC = 0,
  int limitKD = 0,
  int limitKE = 0,
  int limitKF = 0,
  int limitKG = 0,
  int limitKH = 0,
  int limitKI = 0,
  int limitKJ = 0,
  int limitKK = 0,
  int limitKL = 0,
  int limitKM = 0,
  int limitKN = 0,
  int limitKO = 0,
  int limitKP = 0,
  int limitKQ = 0,
  int limitKR = 0,
  int limitKS = 0,
  int limitKT = 0,
  int limitKU = 0,
  int limitKV = 0,
  int limitKW = 0,
  int limitKX = 0,
  int limitKY = 0,
  int limitKZ = 0,
  int limitLA = 0,
  int limitLB = 0,
  int limitLC = 0,
  int limitLD = 0,
  int limitLE = 0,
  int limitLF = 0,
  int limitLG = 0,
  int limitLH = 0,
  int limitLI = 0,
  int limitLJ = 0,
  int limitLK = 0,
  int limitLL = 0,
  int limitLM = 0,
  int limitLN = 0,
  int limitLO = 0,
  int limitLP = 0,
  int limitLQ = 0,
  int limitLR = 0,
  int limitLS = 0,
  int limitLT = 0,
  int limitLU = 0,
  int limitLV = 0,
  int limitLW = 0,
  int limitLX = 0,
  int limitLY = 0,
  int limitLZ = 0,
  int limitMA = 0,
  int limitMB = 0,
  int limitMC = 0,
  int limitMD = 0,
  int limitME = 0,
  int limitMF = 0,
  int limitMG = 0,
  int limitMH = 0,
  int limitMI = 0,
  int limitMJ = 0,
  int limitMK = 0,
  int limitML = 0,
  int limitMM = 0,
  int limitMN = 0,
  int limitMO = 0,
  int limitMP = 0,
  int limitMQ = 0,
  int limitMR = 0,
  int limitMS = 0,
  int limitMT = 0,
  int limitMU = 0,
  int limitMV = 0,
  int limitMW = 0,
  int limitMX = 0,
  int limitMY = 0,
  int limitMZ = 0,
  int limitNA = 0,
  int limitNB = 0,
  int limitNC = 0,
  int limitND = 0,
  int limitNE = 0,
  int limitNF = 0,
  int limitNG = 0,
  int limitNH = 0,
  int limitNI = 0,
  int limitNJ = 0,
  int limitNK = 0,
  int limitNL = 0,
  int limitNM = 0,
  int limitNN = 0,
  int limitNO = 0,
  int limitNP = 0,
  int limitNQ = 0,
  int limitNR = 0,
  int limitNS = 0,
  int limitNT = 0,
  int limitNU = 0,
  int limitNV = 0,
  int limitNW = 0,
  int limitNX = 0,
  int limitNY = 0,
  int limitNZ = 0,
  int limitOA = 0,
  int limitOB = 0,
  int limitOC = 0,
  int limitOD = 0,
  int limitOE = 0,
  int limitOF = 0,
  int limitOG = 0,
  int limitOH = 0,
  int limitOI = 0,
  int limitOJ = 0,
  int limitOK = 0,
  int limitOL = 0,
  int limitOM = 0,
  int limitON = 0,
  int limitOO = 0,
  int limitOP = 0,
  int limitOQ = 0,
  int limitOR = 0,
  int limitOS = 0,
  int limitOT = 0,
  int limitOU = 0,
  int limitOV = 0,
  int limitOW = 0,
  int limitOX = 0,
  int limitOY = 0,
  int limitOZ = 0,
  int limitPA = 0,
  int limitPB = 0,
  int limitPC = 0,
  int limitPD = 0,
  int limitPE = 0,
  int limitPF = 0,
  int limitPG = 0,
  int limitPH = 0,
  int limitPI = 0,
  int limitPJ = 0,
  int limitPK = 0,
  int limitPL = 0,
  int limitPM = 0,
  int limitPN = 0,
  int limitPO = 0,
  int limitPP = 0,
  int limitPQ = 0,
  int limitPR = 0,
  int limitPS = 0,
  int limitPT = 0,
  int limitPU = 0,
  int limitPV = 0,
  int limitPW = 0,
  int limitPX = 0,
  int limitPY = 0,
  int limitPZ = 0,
  int limitQA = 0,
  int limitQB = 0,
  int limitQC = 0,
  int limitQD = 0,
  int limitQE = 0,
  int limitQF = 0,
  int limitQG = 0,
  int limitQH = 0,
  int limitQI = 0,
  int limitQJ = 0,
  int limitQK = 0,
  int limitQL = 0,
  int limitQM = 0,
  int limitQN = 0,
  int limitQO = 0,
  int limitQP = 0,
  int limitQQ = 0,
  int limitQR = 0,
  int limitQS = 0,
  int limitQT = 0,
  int limitQU = 0,
  int limitQV = 0,
  int limitQW = 0,
  int limitQX = 0,
  int limitQY = 0,
  int limitQZ = 0,
  int limitRA = 0,
  int limitRB = 0,
  int limitRC = 0,
  int limitRD = 0,
  int limitRE = 0,
  int limitRF = 0,
  int limitRG = 0,
  int limitRH = 0,
  int limitRI = 0,
  int limitRJ = 0,
  int limitRK = 0,
  int limitRL = 0,
  int limitRM = 0,
  int limitRN = 0,
  int limitRO = 0,
  int limitRP = 0,
  int limitRQ = 0,
  int limitRR = 0,
  int limitRS = 0,
  int limitRT = 0,
  int limitRU = 0,
  int limitRV = 0,
  int limitRW = 0,
  int limitRX = 0,
  int limitRY = 0,
  int limitRZ = 0,
  int limitSA = 0,
  int limitSB = 0,
  int limitSC = 0,
  int limitSD = 0,
  int limitSE = 0,
  int limitSF = 0,
  int limitSG = 0,
  int limitSH = 0,
  int limitSI = 0,
  int limitSJ = 0,
  int limitSK = 0,
  int limitSL = 0,
  int limitSM = 0,
  int limitSN = 0,
  int limitSO = 0,
  int limitSP = 0,
  int limitSQ = 0,
  int limitSR = 0,
  int limitSS = 0,
  int limitST = 0,
  int limitSU = 0,
  int limitSV = 0,
  int limitSW = 0,
  int limitSX = 0,
  int limitSY = 0,
  int limitSZ = 0,
  int limitTA = 0,
  int limitTB = 0,
  int limitTC = 0,
  int limitTD = 0,
  int limitTE = 0,
  int limitTF = 0,
  int limitTG = 0,
  int limitTH = 0,
  int limitTI = 0,
  int limitTJ = 0,
  int limitTK = 0,
  int limitTL = 0,
  int limitTM = 0,
  int limitTN = 0,
  int limitTO = 0,
  int limitTP = 0,
  int limitTQ = 0,
  int limitTR = 0,
  int limitTS = 0,
  int limitTT = 0,
  int limitTU = 0,
  int limitTV = 0,
  int limitTW = 0,
  int limitTX = 0,
  int limitTY = 0,
  int limitTZ = 0,
  int limitUA = 0,
  int limitUB = 0,
  int limitUC = 0,
  int limitUD = 0,
  int limitUE = 0,
  int limitUF = 0,
  int limitUG = 0,
  int limitUH = 0,
  int limitUI = 0,
  int limitUJ = 0,
  int limitUK = 0,
  int limitUL = 0,
  int limitUM = 0,
  int limitUN = 0,
  int limitUO = 0,
  int limitUP = 0,
  int limitUQ = 0,
  int limitUR = 0,
  int limitUS = 0,
  int limitUT = 0,
  int limitUU = 0,
  int limitUV = 0,
  int limitUW = 0,
  int limitUX = 0,
  int limitUY = 0,
  int limitUZ = 0,
  int limitVA = 0,
  int limitVB = 0,
  int limitVC = 0,
  int limitVD = 0,
  int limitVE = 0,
  int limitVF = 0,
  int limitVG = 0,
  int limitVH = 0,
  int limitVI = 0,
  int limitVJ = 0,
  int limitVK = 0,
  int limitVL = 0,
  int limitVM = 0,
  int limitVN = 0,
  int limitVO = 0,
  int limitVP = 0,
  int limitVQ = 0,
  int limitVR = 0,
  int limitVS = 0,
  int limitVT = 0,
  int limitVU = 0,
  int limitVV = 0,
  int limitVW = 0,
  int limitVX = 0,
  int limitVY = 0,
  int limitVZ = 0,
  int limitWA = 0,
  int limitWB = 0,
  int limitWC = 0,
  int limitWD = 0,
  int limitWE = 0,
  int limitWF = 0,
  int limitWG = 0,
  int limitWH = 0,
  int limitWI = 0,
  int limitWJ = 0,
  int limitWK = 0,
  int limitWL = 0,
  int limitWM = 0,
  int limitWN = 0,
  int limitWO = 0,
  int limitWP = 0,
  int limitWQ = 0,
  int limitWR = 0,
  int limitWS = 0,
  int limitWT = 0,
  int limitWU = 0,
  int limitWV = 0,
  int limitWW = 0,
  int limitWX = 0,
  int limitWY = 0,
  int limitWZ = 0,
  int limitXA = 0,
  int limitXB = 0,
  int limitXC = 0,
  int limitXD = 0,
  int limitXE = 0,
  int limitXF = 0,
  int limitXG = 0,
  int limitXH = 0,
  int limitXI = 0,
  int limitXJ = 0,
  int limitXK = 0,
  int limitXL = 0,
  int limitXM = 0,
  int limitXN = 0,
  int limitXO = 0,
  int limitXP = 0,
  int limitXQ = 0,
  int limitXR = 0,
  int limitXS = 0,
  int limitXT = 0,
  int limitXU = 0,
  int limitXV = 0,
  int limitXW = 0,
  int limitXX = 0,
  int limitXY = 0,
  int limitXZ = 0,
  int limitYA = 0,
  int limitYB = 0,
  int limitYC = 0,
  int limitYD = 0,
  int limitYE = 0,
  int limitYF = 0,
  int limitYG = 0,
  int limitYH = 0,
  int limitYI = 0,
  int limitYJ = 0,
  int limitYK = 0,
  int limitYL = 0,
  int limitYM = 0,
  int limitYN = 0,
  int limitYO = 0,
  int limitYP = 0,
  int limitYQ = 0,
  int limitYR = 0,
  int limitYS = 0,
  int limitYT = 0,
  int limitYU = 0,
  int limitYV = 0,
  int limitYW = 0,
  int limitYX = 0,
  int limitYY = 0,
  int limitYZ = 0,
  int limitZA = 0,
  int limitZB = 0,
  int limitZC = 0,
  int limitZD = 0,
  int limitZE = 0,
  int limitZF = 0,
  int limitZG = 0,
  int limitZH = 0,
  int limitZI = 0,
  int limitZJ = 0,
  int limitZK = 0,
  int limitZL = 0,
  int limitZM = 0,
  int limitZN = 0,
  int limitZO = 0,
  int limitZP = 0,
  int limitZQ = 0,
  int limitZR = 0,
  int limitZS = 0,
  int limitZT = 0,
  int limitZU = 0,
  int limitZV = 0,
  int limitZW = 0,
  int limitZX = 0,
  int limitZY = 0,
  int limitZZ = 0,
  int limitAA = 0,
  int limitAB = 0,
  int limitAC = 0,
  int limitAD = 0,
  int limitAE = 0,
  int limitAF = 0,
  int limitAG = 0,
  int limitAH = 0,
  int limitAI = 0,
  int limitAJ = 0,
  int limitAK = 0,
  int limitAL = 0,
  int limitAM = 0,
  int limitAN = 0,
  int limitAO = 0,
  int limitAP = 0,
  int limitAQ = 0,
  int limitAR = 0,
  int limitAS = 0,
  int limitAT = 0,
  int limitAU = 0,
  int limitAV = 0,
  int limitAW = 0,
  int limitAX = 0,
  int limitAY = 0,
  int limitAZ = 0,
  int limitBA = 0,
  int limitBB = 0,
  int limitBC = 0,
  int limitBD = 0,
  int limitBE = 0,
  int limitBF = 0,
  int limitBG = 0,
  int limitBH = 0,
  int limitBI = 0,
  int limitBJ = 0,
  int limitBK = 0,
  int limitBL = 0,
  int limitBM = 0,
  int limitBN = 0,
  int limitBO = 0,
  int limitBP = 0,
  int limitBQ = 0,
  int limitBR = 0,
  int limitBS = 0,
  int limitBT = 0,
  int limitBU = 0,
  int limitBV = 0,
  int limitBW = 0,
  int limitBX = 0,
  int limitBY = 0,
  int limitBZ = 0,
  int limitCA = 0,
  int limitCB = 0,
  int limitCC = 0,
  int limitCD = 0,
  int limitCE = 0,
  int limitCF = 0,
  int limitCG = 0,
  int limitCH = 0,
  int limitCI = 0,
  int limitCJ = 0,
  int limitCK = 0,
  int limitCL = 0,
  int limitCM = 0,
  int limitCN = 0,
  int limitCO = 0,
  int limitCP = 0,
  int limitCQ = 0,
  int limitCR = 0,
  int limitCS = 0,
  int limitCT = 0,
  int limitCU = 0,
  int limitCV = 0,
  int limitCW = 0,
  int limitCX = 0,
  int limitCY = 0,
  int limitCZ = 0,
  int limitDA = 0,
  int limitDB = 0,
  int limit
```