

```

double Martingale(double Base_Lots, double Multiplier, int Max)
{
  int Loss_Count = 0;
  for(int i = OrdersHistoryTotal()-1; i >= 0; i--)
  {
    if(OrderSelect(i,SELECT_BY_POS,MODE_HISTORY) == true)
    {
      if(OrderMagicNumber() == Magic && OrderSymbol() == Symbol())
      {
        if(OrderProfit() <= 0)
        {
          Loss_Count++;
        }
        else if(OrderProfit() > 0)
        {
          break;
        }
      }
    }
  }

  if(Loss_Count >= Max)
  {
    Loss_Count = Max;
  }

  double Lots = Base_Lots * MathPow(Multiplier, Loss_Count);
  if(Lots >= MarketInfo(Symbol(),MODE_MAXLOT))
  {
    Lots = MarketInfo(Symbol(),MODE_MAXLOT);
  }

  return(Lots);
}

```

The image shows a C++ code snippet for a Martingale trading strategy. Red annotations highlight specific parts of the code:

- ア** (A) points to the function parameters: `double Base_Lots, double Multiplier, int Max`.
- イ** (I) points to the `Loss_Count` variable declaration: `int Loss_Count = 0;`.
- ウ** (U) points to the `OrdersHistoryTotal()-1` expression in the for loop: `OrdersHistoryTotal()-1`.
- エ** (E) points to the `OrderProfit() <= 0` condition in the inner if statement.
- オ** (O) points to the `break;` statement in the else-if block.
- カ** (Ka) points to the `MathPow(Multiplier, Loss_Count)` expression in the `Lots` calculation.