The library supports the customization of the progress function.

The progress function is called internally after each iteration and is a good place where to put custom code for various purposes:

- custom control value output;
- intermediary computations (e.g. external validation of results every k steps)
- ...

The following code snippet is available from examples/sample-code-pfunc.cc and demonstrate how to override the default progress function in order to print out the cost of an iteration in ms, every 1000 iterations:

```
#include "cmaes.h"
#include <iostream>

using namespace libcmaes;

FitFunc rosenbrock = [](const double *x, const int N)
{
  double val = 0.0;
  for (int i=0;i<N-1;i++)
    {
      val += 100.0*pow((x[i+1]-x[i]*x[i]),2) + pow((x[i]-1.0),2);
    }
  return val;
};

ProgressFunc<CMAParameters<>,CMASolutions> select_time = [](const CMAParameters<> &cmaparams, const CMASol
{
  if (cmasols._niter % 1000 == 0)
    std::cerr << cmasols._elapsed_last_iter << std::endl;
  return 0;
};

int main(int argc, char *argv[])
{
  int dim = 100; // problem dimensions.
  std::vector<double> x0(dim,10.0);
  double sigma = 0.1;
  //int lambda = 100; // offsprings at each generation.
  CMAParameters<> cmaparams(dim,&x0.front(),sigma);
  //cmaparams._algo = BIPOP_CMAES;
  CMASolutions cmasols = cmaes<>(rosenbrock,cmaparams,select_time);
  std::cout << "best solution: " << cmasols << std::endl;
  std::cout << "optimization took " << cmasols._elapsed_time / 1000.0 << " seconds\n";
  retrn cmasols._run_status;
}
```