```cpp
#pragma once

#include <bitset>
#include <random>
#include <string>
#include <algorithm>
#include <chrono>

#include "NN.h"
#include "SoilMathTypes.h"
#include "MathException.h"

namespace SoilMath
{

    class GA
    {
    public:
        GA();
        GA(NNfunctionType nnfunction, uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons);
        ~GA();

        void Evolve(const ComplexVect_t &inputValues, Weight_t &weights, std::vector<Weight_t> &prevWeights, MinMaxWeight_t rangeweights, Predict_t goal, uint32_t maxGenerations = 200, uint32_t popSize = 30);
        void Evolve(const InputLearnVector_t &inputValues, Weight_t &weights, MinMaxWeight_t rangeweights, OutputLearnVector_t &goal, uint32_t maxGenerations = 200, uint32_t popSize = 30);

    private:
        NNfunctionType NNfuction;
        uint32_t inputneurons;
        uint32_t hiddenneurons;
        uint32_t outputneurons;

        Population_t Genesis(const Weight_t &weights, MinMaxWeight_t rangeweights, uint32_t popSize);
        void CrossOver(Population_t &pop);
        void Mutate(Population_t &pop);
        void GrowToAdulthood(Population_t &pop, const ComplexVect_t &inputValues, MinMaxWeight_t rangeweights, Predict_t goal, float &totalFitness);
        void GrowToAdulthood(Population_t &pop, const InputLearnVector_t &inputValues, MinMaxWeight_t rangeweights, OutputLearnVector_t &goal, float &totalFitness);
        bool SurvivalOfTheFittest(Population_t &pop, float &totalFitness);
```

```
38
39          static bool PopMemberSort(PopMember_t i, PopMember_t j) { return (i.Fitness < j.Fitness); }
40
41          template <typename T>
42          inline Genome_t ConvertToGenome(T value, std::pair<T, T> range)
43          {
44              uint32_t intVal = static_cast<uint32_t>((UINT32_MAX * (range.first + value)) / (range.second - range.first));
45              Genome_t retVal(intVal);
46              return retVal;
47          };
48
49          template <typename T>
50          inline T ConvertToValue(Genome_t gen, std::pair<T, T> range)
51          {
52              T retVal = range.first + (((range.second - range.first) * static_cast<T>(gen.to_ulong())) / UINT32_MAX);
53              return retVal;
54          };
55      };
56
57  }
```