CLASS VECTOR REFERENCE MANUAL

RETURN	METHOD NAME	PARAMETERS	DESCRIPTION
		INITIALIZATION	
	Vector	void	New Vector
	Vector	<T>* X int N	Convert array *X* of size *N* to a vector
	Vector	int Length	New Vector of start dimension *Length*
Vector<T>*	Clone	void	Copy of a vector
int	GetLength	void	Get length of the vector
int	GetStepSize	void	Get amount of new memory allocated when vector is full
void	SetStepSize	int StepSize	Set amount of new memory allocated when vector is full
T	GetValue	int Index	Get an array element
		ADD / REMOVE METHODS	
void	Clear	void	Remove all elements from the vector
void	Add	<T> X	Add a new element in the vector
void	AddAt	<T> X int Index	Add a new element in the vector on *Index* position
void	RemoveAt	int Index	Remove element on *Index* position from the vector
Vector<T>*	Extract	int FromIndex int ToIndex	Build a new vector with elements from position *FromIndex* to *ToIndex*
		PRE-BUILT VECTORS	
Vector<double>*	ZeroVector	int Length	Build a new vector of zeros with *Length* elements
Vector<double>*	RandVector	int Length	Build a new vector with *Length* random elements
Vector<T>*	GetSequence	<T> Start <T> Step <T> End	Build a new vector with elements from *Start* to *End*, with step *Step*
		MATHEMATICAL METHODS	
void	SumScalar	<T> X	Sum *X* to each vector element
void	ProductScalar	<T> X	Multiply *X* to each vector element
void	DivideScalar	<T> X	Divide *X* to each vector element
void	PowScalar	<T> X	Pow *X* to each vector element
void	SumVector	Vector<T>* V	Sum vector with another one
Vector<T>*	SumVector	Vector<T>* V1 Vector<T>* V2	Sum vectors *V1* and *V2*
void	SubtractVector	Vector<T>* V	Subtract vector with another one
Vector<T>*	SubtractVector	Vector<T>* V1 Vector<T>* V2	Subtract vectors *V1* and *V2*
void	ProductVector	Vector<T>* V	Multiply vector with another one
Vector<T>*	ProductVector	Vector<T>* V1 Vector<T>* V2	Multiply vectors *V1* and *V2*
<T>	ProductVectorScalar	Vector<T>* V	Scalar product of two vectors
<T>	ProductVectorScalar	Vector<T>* V1 Vector<T>* V2	Scalar product of vectors *V1* and *V2*
<T>	Sum	void	Sum all vector elements
<T>	AbsSum	void	Sum all vector absolute values

CLASS VECTOR REFERENCE GUIDE

RETURN	METHOD NAME	PARAMETERS	DESCRIPTION
COMPARISON METHODS			
<T>	Min	void	Min vector value
void	Min	<T>* MinValue int* MinIndex	Min vector value and position
<T>	MinAbs	void	Min vector absolute value
void	MinAbs	<T>* MinValue int* MinIndex	Min vector absolute value and position
<T>	Max	void	Max vector value
void	Max	<T>* MinValue int* MinIndex	Max vector value and position
<T>	MaxAbs	void	Max vector absolute value
void	MaxAbs	<T>* MinValue int* MinIndex	Max vector absolute value and position
<T>	Mean	void	Mean vector value
<T>	MeanAbs	void	Mean vector absolute value
SORTING METHODS			
void	Sort	void	Sort vector
void	RemoveDuplicates	void	Remove duplicates from vector
int	Find	<T> X	Find position of X in the vector
INPUT / OUTPUT METHODS			
Vector<double>*	Load	char* Filename	Load vector from file
void	Save	char* Filename	Save vector to file
void	Print	void	Print vector to output
void	Print	char* VectorName	Print vector to output

CLASS VECTOR EXAMPLES

```
#include "Vector.h"

using namespace onlinesvr;


int main ()
{
	// Make a new vector
	Vector<int>* V1 = new Vector<int>();

	// Fill vector
	for (int i=0; i<5; i++) {
		V1->Add(i);
	}

	// Add 5 to each vector element
	V1->SumScalar(5);

	// Copy vector
	Vector<int>* V2 = V1->Clone();

	// Normalize vector
	int MaxValue = V2->Max();
	V2->DivideScalar(MaxValue);

	// Print vectors
	V1->Print("V1");
	V2->Print("V2");

	// Vector sum
	Vector<int>* V3 = V1->SumVector(V1,V2);
	V3->Print("V3");

	// Rand Vector of 4 elements
	Vector<double>* V4 = Vector<double>::RandVector(4);
	V4->Print("V4");

	// Sort
	V4->Sort();
	V4->Print("Vsort");

	// Save vector
	V4->Save("V4.vec");

	// Delete vectors
	delete V1;
	delete V2;
	delete V3;
	delete V4;
}
```