Details :

{

Title : "Example" ,

Author : "Somebody" ,

File Name : "Example.mcsource" , //Generated Line, cannot be edited.

Date Created : "2-15-2019" , //Generated Line, cannot be edited.

Last Modified: "2-15-2019" , //Generated Line, cannot be edited.

Description : "An example of an mcsource file for the M.A.S. Language."

};

Context (NotationEngine::Core::Launch);

Using :

{

StandardLibrary::Compiler::ConsoleIO ::CoutStream::Linked,

StandardLibrary::Compiler::DerrivedTypes ::Strings ::String ::Linked,

StandardLibrary::Compiler::DerrivedTypes ::Strings ::ToString ::Linked,

StandardLibrary::Compiler::Execution ::ResultCode::Linked,

StandardLibrary::Compiler::FundamentalTypes ::UInt64 ::Linked,

StandardLibrary::Compiler::Memory::Pointers :

{

Address ::Linked,

Data ::Linked,

Pointer ::Linked

},

StandardLibrary::Compiler::TypeDeduction :

{

IsFundamental::Linked

}

};

Module (Compiler.Level.4) : //Equivalent to the abstraction level of C++.

{

Alias (Standard.Version.2019) :

{

Implicit : UseAutoCamelContext;

Append :  
 {

//Standard Library

type Cout : SL::C::CIO::CS::CoutStream,

type ResultCode : SL::C::E ::ResultCode,

type Pointer<> : SL::C::M ::P ::Pointer<> ,

type String : SL::C::DT ::S ::String ,

type UInt32 : SL::C::FT ::UInt64 ,

type :

{

Cout : SL::C::CIO::CS::CoutStream,

String : SL::C::DT ::S ::String

},

function Address : SL::C::M ::P::Address ,

function Data : SL::C::M ::P::Data ,

function IsFundamental : SL::C::TD ::IsFundamental,

function ToString : SL::C::DT::S::ToString

};

Source (Version.1) :

{

//Generics

concept FundamentalType :

{

type typeID : deduce IsFundamental(typeID)::Result

};

//Declarations

declare :

{

firstInt : static constant UInt64,

secondInt : static constant UInt64

};

declare additionResult : dynamic variable Pointer<UInt64>;

//Assignments

additionResult = Allocate<UInt64>;

//Functions

function :

{

AddFundamentalType

(

type \_numOne,

\_numTwo :

immutable FundamentalType

)

-> FundamentalType :

{

declare result : auto FundamentalType;

result = \_numOne + \_numTwo;

return result;

};

PrintFundamentalType(type \_numToPrint : immutable Pointer<FundamentalType>) -> void :

{

Cout << "Value: " << ToString(Data(\_numToPrint) << "\n";

return;

}

};

function PrintStartMsg(void) -> void :

{

Cout << "Hello World!" << "\n";

return;

};

entryPoint execution(void) -> ResultCode :

{

PrintStartMsg();

Data(additionResult) = AddFundamentalType(firstInt, secondInt);

PrintFundementalType(additionResult);

return ResultCode::Success;

};

}; //EndSource

}; //EndAlias

}; //EndModule