B. C++ Syntax Reference Guide

Name	Syntax	Example

Pre-processor directives	`#include <libraryName>`
	`#define <MACRO_NAME> <expansion>`
		`#include <iostream> // for CIN & COUT`
		`#include <iomanip>  // for setw()`
		`#include <fstream>  // for IFSTREAM`
		`#include <string>   // for STRING`
		`#include <cctype>   // for ISUPPER`
		`#include <cstring>  // for STRLEN`
		`#include <cstdlib>  // for ATOF`
		
		`#define PI        3.14159`
		`#define LANGUAGE  "C++"`

Function	`<ReturnType> <functionName>(<params>)`
	`{`
	`  <statements>`
	`  return <value>;`
	`}`
		`int main()`
		`{`
		`  cout << "Hello world\n";`
		`  return 0;`
		`}`

Function parameters	`<DataType> <passByValueVariable>,`
	`<DataType> & <passByReferenceVariable>,`
	`const <DataType> <CONSTANT_VARIABLE>,`
	`<BaseType> <arrayVariable>[]`
		`void function(int value,`
		`  int &reference,`
		`  const int CONSTANT,`
		`  int array[])`
		`{`
		`}`

COUT	`cout << <expression> << … ;`	`cout << "Text in quotes"`
		`  << 6 * 7`
		`  << getNumber()`
		`  << endl;`

Formatting output for money	`cout.setf(ios::fixed);`
	`cout.setf(ios::showpoint);`
	`cout.precision(<integerExpression>);`
		`cout.setf(ios::fixed);`
		`cout.setf(ios::showpoint);`
		`cout.precision(2);`

Declaring variables	`<DataType> <variableName>;`
	`<DataType> <variableName> = <init>;`
	`const <DataType> <VARIABLE_NAME>;`
		`int integerValue;`
		`float realNumber = 3.14159;`
		`const char LETTER_GRADE = 'A';`

CIN	`cin >> <variableName>;`	`cin >> variableName;`

IF statement	`if (<Boolean-expression>)`
	`{`
	`  <statements>`
	`}`
	`else`
	`{`
	`  <statements>`
	`}`
		`if (grade >= 70.0)`
		`  cout << "Great job!\n";`
		`else`
		`{`
		`  cout << "Try again.\n";`
		`  pass = false;`
		`}`

Asserts	`assert(<Boolean-expression>);`	`#include <cassert> // at top of file`
		`{`
		`  assert(gpa >= 0.0);`
		`}`

Name	Syntax	Example

FOR loop	for (<initialization statement>;
	  <Boolean-expression>;
	  <increment statement>)
	{
	  <statements>
	}
		for (int iList = 0;
		  iList < sizeList;
		  iList++)
		  cout << list[iList];

WHILE loop	while (<Boolean-expression>)
	{
	  <statements>
	}
		while (input <= 0)
		  cin >> input;

DO-WHILE Loop	do
	{
	  <statements>
	}
	while (<Boolean-expression>);
		do
		  cin >> input;
		while (input <= 0);

Read from File	ifstream <streamVar>(<fileName>);
	if (<streamVar>.fail())
	{
	  <statements>
	}
	
	<streamVar> >> <variableName>;
	
	<streamVar>.close();
		#include <fstream> // at top of file
		{
		  ifstream fin("data.txt");
		  if (fin.fail())
		  return false;
		
		  fin >> value;
		
		  fin.close();
		}

Write to File	ofstream <streamVar>(<fileName>);
	if (<streamVar>.fail())
	{
	  <statements>;
	}
	
	<streamVar> << <expression>;
	
	<streamVar>.close();
		#include <fstream> // at top of file
		{
		  ofstream fout("data.txt");
		  if (fout.fail())
		  return false;
		
		  fout << value << endl;
		
		  fout.close();
		}

Fill an array	<BaseType> <arrayName>[<size>];
	<BaseType> <arrayName>[] =
	  { <CONST_1>, <CONST_2>, … };
	
	for (int i = 0; i < <size>; i++)
	  <arrayName>[i] = <expression>;
		int grades[10];
		for (int i = 0; i < 10; i++)
		{
		  cout << "Grade " << i + 1 << ": ";
		  cin >> grades[i];
		}

C-Strings	char <stringName>[<size>];
	cin >> <stringName>;
	for (char *<ptrName> = <stringName>;
	  *<ptrName>;
	  <ptrName>++)
	  cout << *<ptrName>;
		char firstName[256];
		cin >> firstName;
		for (char *p = firstName; *p; p++)
		  cout << *p;

String Class	string <stringName>;
	cin >> <stringName>;
	cout << <stringName>;
	getline(<streamName>, <stringName>);
	
	if (<stringName1> == <stringName2>)
	  <statemement>;
	
	<stringName1> += <stringName2>;
	<stringName1> = <stringName2>;
		string string1; // declare
		string string2 = "124"; // initialize
		
		cin >> string1; // input
		getline(cin, string2); // getline
		if (string1 == string2) // compare
		  string1 += string2; // append
		string2 = string1; // copy

Name	Syntax	Example

Switch	switch (<integer-expression>)
	{
	  case <integer-constant>:
	  <statements>
	  break; // optional
	  …
	  default: // optional
	  <statements>
	}
		switch (value)
		{
		  case 3:
		  cout << "Three";
		  break;
		  case 2:
		  cout << "Two";
		  break;
		  case 1:
		  cout << "One";
		  break;
		  default:
		  cout << "None!";
		}

Conditional Expression	<Boolean-expression> ? <expression> : <expression>	cout << "Hello, " << (isMale ? "Mr. " : "Mrs. ") << lastName;

Multi-dimensional array	<BaseType> <arrayName>[<SIZE>][<SIZE>];
	<BaseType> <arrayName>[][<SIZE>] =
	  {
	  { <CONST_0_0>, <CONST_0_1>, … },
	  { <CONST_1_0>, <CONST_1_1>, … },
	  …
	  };
	
	<arrayName>[<index>][<index>] =
	  <expression>;
		int board[3][3];
		
		for (int row = 0; row < 3; row++)
		  for (int col = 0; col < 3; col++)
		  board[row][col] = 10;

Allocate memory	<ptr> = new(nothrow) <DataType>;
	<ptr> = new(nothrow> <DataType>(<init>);
	<ptr> = new(nothrow) <BaseType>[<SIZE>];
		float *pNum1 = new(nothrow) float;
		int *pNum2 = new(nothrow) int(42);
		char *text = new(nothrow) char[256];

Free memory	delete <pointer>; // one value
	delete [] <arrayPointer>; // an array
		delete pNumber;
		delete [] text;

Command line parameters	int main(int <countVariable>,
	  char **<arrayVariable>)
	{
	}
		int main(int argc, char **argv)
		{
		}

Input errors	<istream>.fail();
	<istream>.clear();
	<istream>.ignore(<numChars>,<token>)
		if (cin.fail())
		{
		  cin.clear();
		  cin.ignore(256, '\n');
		}

Exceptions	try
	{
	  <statements>
	  throw <expression>
	}
	catch (<declaration>)
	{
	  <statements>
	}
		try
		{
		  cin >> data;
		  if (data < 0)
		  throw data;
		}
		catch (int error)
		{
		  cout << "An error occurred!";
		}

Structures	struct <DataType>
	{
	  <member variable declarations>
	};
		struct Point
		{
		  int row;
		  int col;
		};

Header files	#ifndef <FILE_TAG>
	#define <FILE_TAG>
	
	<body of the header>
	
	#endif // FILE_TAG
		#ifndef _POINT_H_
		#define _POINT_H_
		
		struct Point
		{
		  int row;
		  int col;
		};
		
		#endif // _POINT_H_

makefile	<target> : <Dependency List>
	  <Recipe>
		a.out: file.o interface.o
		  g++ file.o interface.o
		  tar –cf file.tar *.h *.cpp

Default parameters	<Return> <funcName>(<DataType> <variable> = <value>);	void func(int value = 0);
Function pointer	<Return> (*<pointer>)(<param list>);	void (*ptrFunction)(int value);

Class syntax	class <ClassTag>
	{
	  <public/private>:
	  <variable declarations>
	  <method definitions>
	};
		class Point
		{
		  public:
		  void set(int r, int c);
		  private:
		  int row;
		  int col;
		};

Method definition	<Return> <ClassName>::<methodName>()
	{
	  <statements>
	}
		void Point :: set (int r, int c)
		{
		  row = r;
		  col = c;
		}