(C++) ROOT & ALICE Data Analysis (++ More ??!!)

Indranil Das

indranil.das@cern.ch

Outline : Life cycle of EHEP PhD student

1. C++ language
2. ROOT : HEP analysis tool
3. AliRoot : ALICE Analysis Software
4. Logistics

In Brief

1. C++ language, C++ class
2. Language Class : Why ?
3. Communication between the human and machine
4. Advance features of **C** and concept of class from **Simula**
5. Tested and evolved in time C++98, C++03, C++11, C++14, C++17,**C++20**, C++23
6. Learn from the inventor, Bjarne Stroustrup :
   https://www.youtube.com/watch?v=JBjjnqG0BP8
   https://www.youtube.com/watch?v=69edOm889V4

In Brief

1. C++ language, C++ class
2. Language Class : Why ?
3. Communication between the human and machine
4. Advance features of **C** and concept of class from **Simula**
5. Tested and evolved in time C++98, C++03, C++11, C++14, C++17,**C++20**, C++23
6. Learn from the inventor, Bjarne Stroustrup :
   https://www.youtube.com/watch?v=JBjjnqG0BP8
   https://www.youtube.com/watch?v=69edOm889V4

In Brief

1. C++ language, C++ class
2. Language Class : Why ?
3. Communication between the human and machine
4. Advance features of **C** and concept of class from **Simula**
5. Tested and evolved in time C++98, C++03, C++11, C++14, C++17,**C++20**, C++23
6. Learn from the inventor, Bjarne Stroustrup :
   https://www.youtube.com/watch?v=JBjjnqG0BP8
   https://www.youtube.com/watch?v=69edOm889V4

In Brief

1. C++ language, C++ class
2. Language Class : Why ?
3. Communication between the human and machine
4. Advance features of **C** and concept of class from **Simula**
5. Tested and evolved in time C++98, C++03, C++11, C++14, C++17,**C++20**, C++23
6. Learn from the inventor, Bjarne Stroustrup :
   https://www.youtube.com/watch?v=JBjjnqG0BP8
   https://www.youtube.com/watch?v=69edOm889V4

In Brief

1. C++ language, C++ class
2. Language Class : Why ?
3. Communication between the human and machine
4. Advance features of **C** and concept of class from **Simula**
5. Tested and evolved in time C++98, C++03, C++11, C++14, C++17,**C++20**, C++23
6. Learn from the inventor, Bjarne Stroustrup :
   https://www.youtube.com/watch?v=JBjjnqG0BP8
   https://www.youtube.com/watch?v=69edOm889V4

In Brief

1. C++ language, C++ class
2. Language Class : Why ?
3. Communication between the human and machine
4. Advance features of **C** and concept of class from **Simula**
5. Tested and evolved in time C++98, C++03, C++11, C++14, C++17,**C++20**, C++23
6. Learn from the inventor, Bjarne Stroustrup :
   https://www.youtube.com/watch?v=JBjjnqG0BP8
   https://www.youtube.com/watch?v=69edOm889V4

Get set go...

- "The C++ Programming Language" Bjarne Stroustrup
- "Object-Oriented Programming with C++" E. Balagurusamy
- http://en.cppreference.com/w/cpp
- http://www.cplusplus.com/doc/tutorial/
- https://www.tutorialspoint.com/cplusplus/
- In Redhat based system : `yum install gcc-c++` , `dnf install gcc-c++` (on newer)
- In Debian based system : `apt-get install g++`

Get set go...

1. Compile : `g++ sqroot.C -o sqroot`
2. Run : `./sqroot`
3. output : `rt : 1.41421`
4. You can also use IDE : Eclipse, Xcode, Visual Studio, NetBeans

Another example...

swap.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using std::cout;

int main()
{
  void swap(int*, int*);
  int a(2), b = 3;
  swap(&a,&b);
  cout<<"a : "<<a<<", b : "<<b<<" \n";
  return true;
}

void swap(int *x, int *y)
{
  int z = *x;
  *x = *y;
  *y = z;
}
```

-:--- swap.C All L2 (C++/l Abbrev)

(No changes need to be saved)

Technical terminology

1. header file
2. namespace
3. scope resolution
4. function declaration
5. data type / class
6. assignment operator
7. function call
8. reference operator
9. insertion operator (same form as bit-shifting operator)
10. c-type end of line
11. function definition
12. dereference operator

C++ basics

1. C++ keywords
2. Identifiers
3. Data Types
4. Scope Resolution
5. Memory Management

C++ keywords

alignas (since C++11)	dynamic_cast	reinterpret_cast
alignof (since C++11)	else	requires (since C++20)
and	enum	return
and_eq	explicit	short
asm	export(1)	signed
atomic_cancel (TM TS)	extern(1)	sizeof(1)
atomic_commit (TM TS)	false	static
atomic_noexcept (TM TS)	float	static_assert (since C++11)
auto(1)	for	static_cast
bitand	friend	struct(1)
bitor	goto	switch
bool	if	synchronized (TM TS)
break	import (modules TS)	template
case	inline(1)	this
catch	int	thread_local (since C++11)
char	long	throw
char16_t (since C++11)	module (modules TS)	true
char32_t (since C++11)	mutable(1)	try
class(1)	namespace	typedef
compl	new	typeid
concept (since C++20)	noexcept (since C++11)	typename
const	not	union
constexpr (since C++11)	not_eq	unsigned
const_cast	nullptr (since C++11)	using(1)
continue	operator	virtual
decltype (since C++11)	or	void
default(1)	or_eq	volatile
delete(1)	private	wchar_t
do	protected	while
double	public	xor
	register(2)	xor_eq

- http://en.cppreference.com/w/cpp/keyword

Identifiers

1. Identifiers : Names of variables, functions, arrays and classes.
2. The rules for naming the identifiers
    1. Alphabetic characters, digits and underscores
    2. Can not start with a digit
    3. Case sensitive
    4. C++ keywords can not be used

Identifiers

1. Identifiers : Names of variables, functions, arrays and classes.
2. The rules for naming the identifiers
    1. Alphabetic characters, digits and underscores
    2. Can not start with a digit
    3. Case sensitive
    4. C++ keywords can not be used

Identifiers

1. Identifiers : Names of variables, functions, arrays and classes.
2. The rules for naming the identifiers
    1. Alphabetic characters, digits and underscores
    2. Can not start with a digit
    3. Case sensitive
    4. C++ keywords can not be used

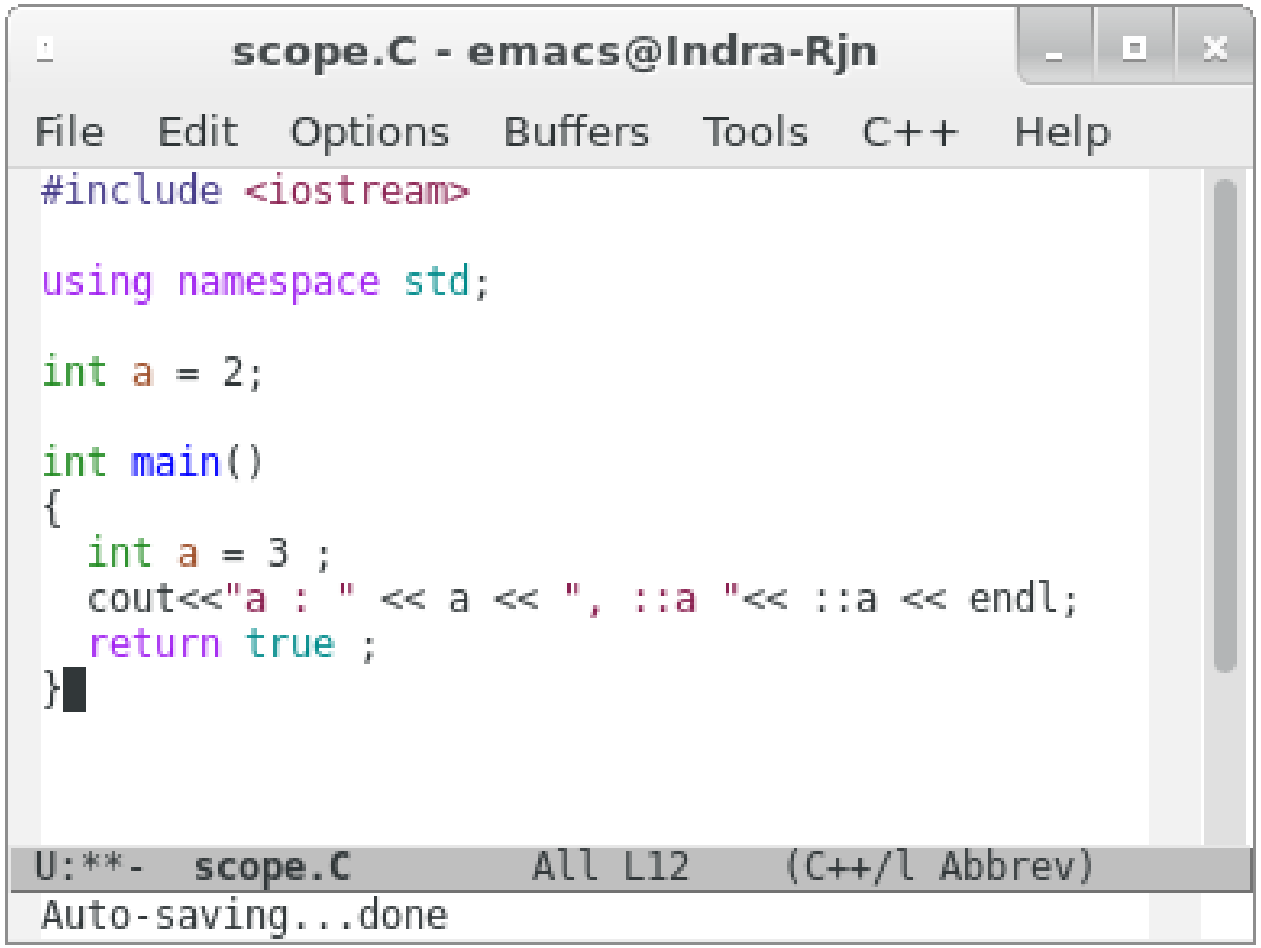
Identifiers

1. Identifiers : Names of variables, functions, arrays and classes.
2. The rules for naming the identifiers
    1. Alphabetic characters, digits and underscores
    2. Can not start with a digit
    3. Case sensitive
    4. C++ keywords can not be used

Data Types

Type	Size (in bytes)	Range
char	1	-127 to 127 or 0 to 255
unsigned char	1	0 to 255
int	4	-2147483648 to 2147483647
unsigned int	4	0 to 4294967295
short int	2	-32768 to 32767
unsigned short int	2	0 to 65,535
long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
float	4	+/- 3.4e +/- 38 (~7 digits)
double	8	+/- 1.7e +/- 308 (~15 digits)

Scope Resolution



Memory Management

memory.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

int main()
{
  int ROW = 2;
  int COL = 3;
  double **pvalue  = new double* [ROW]; // Allocate memory for rows

  // Now allocate memory for columns
  for(int i = 0; i < ROW; i++) {
    pvalue[i] = new double[COL];
  }

  for(int i = 0; i < ROW; i++) {
    delete[] pvalue[i];
  }
  delete [] pvalue;

  return true;
}
```

U:--- memory.C All L4 (C++/l Abbrev)

- https://www.tutorialspoint.com/cplusplus/cpp_dynamic_memory.htm

C++ class terminology

- Encapsulation : The wrapping up of data and functions into a class is known as encapsulation.
- Abstraction : The abstraction is the act of revealing the essential features of class to the users without disclosing the processing inside the class.
- Data member : The variables of the class are known as data member.
- Methods of Member Functions : The functions which processes the data members are called methods or member functions

Features C++ class

- Access modifiers (default is private)
- Constructor and Destructor
- Copy Constructor
- Friend Function
- "this" pointer
- Static Member
- Inheritance : Base and Derived classes
- Polymorphism
- Operator overloading

box.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Box {
   public:
      double length;   // Length of a box
      double breadth;  // Breadth of a box
      double height;   // Height of a box
};

int main() {
  Box Box1;        // Declare Box1 of type Box
  double volume = 0.0;     // Store the volume of a box here

   // box 1 specification
  Box1.height = 5.0;
  Box1.length = 6.0;
  Box1.breadth = 7.0;

  // volume of box 1
  volume = Box1.height * Box1.length * Box1.breadth;
  cout << "Volume of Box1 : " << volume <<endl;

  return true;
}
```

U:--- box.C All L2 (C++/l Abbrev)

- https://www.tutorialspoint.com/cplusplus/

line.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Line {
public:
  void setLength( double len = 0) {length = len ;}
  double getLength( void );
  Line() {cout << "Object is being created" << endl;}
  ~Line() {cout << "Object is being deleted" << endl;}

private:
  double length;
};

double Line::getLength( void ) {
  return length;
}

int main( ) {

  for(int i=0;i<4;i++){
    Line line;
    line.setLength(6.0*i);
    cout << "Length of line : " << line.getLength() <<endl;
  }

  return true;
}
```

U:--- **line.C** Top L2 (C++/l Abbrev)

(No changes need to be saved)

line1.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Line {
public:
  void setLength( double len = 0) {length = len ;}
  double getLength( void );
  Line( const Line &obj) { length = obj.length;}
  Line() {cout << "Object is being created" << endl;}
  ~Line() {cout << "Object is being deleted" << endl;}

private:
  double length;
};

double Line::getLength( void ) {
  return length;
}

int main( ) {
  Line line1;
  line1.setLength(6.0);

  Line line2(line1);
  Line line3;
  line3 = line2;

  cout << "Length of line1 : " << line1.getLength() <<endl;
  cout << "Length of line2 : " << line2.getLength() <<endl;
  cout << "Length of line3 : " << line3.getLength() <<endl;

  return true;
}
```

-:--- **line1.C** Top L20 (C++/l Abbrev)

line1.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Line {
public:
  void setLength( double len = 0) {length = len ;}
  double getLength( void );
  //Line( const Line &obj) { length = obj.length;}
  Line() {cout << "Object is being created" << endl;}
  ~Line() {cout << "Object is being deleted" << endl;}

private:
  double length;
};

double Line::getLength( void ) {
  return length;
}

int main( ) {
  Line line1;
  line1.setLength(6.0);

  Line line2(line1);
  Line line3;
  line3 = line2;

  cout << "Length of line1 : " << line1.getLength() <<endl;
  cout << "Length of line2 : " << line2.getLength() <<endl;
  cout << "Length of line3 : " << line3.getLength() <<endl;

  return true;
}
```

-:--- **line1.C** Top L2 (C++/l Abbrev)

friend.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Line {
public:
  void setLength( double len = 0) {length = len ;}
  double getLength( void ) {return length;}
  Line() {cout << "Object is being created" << endl;}
  ~Line() {cout << "Object is being deleted" << endl;}

  friend void timesTwo( Line line);

private:
  double length;
};

void timesTwo(Line line){
  line.length *= 2.0;
  cout << "Length of line : " << line.getLength() <<endl;
}

int main( ) {

  Line line;
  line.setLength(6.0);
  timesTwo(line);

  return true;
}
```

-:--- **friend.C** All L4 (C++/l Abbrev)

Wrote /Data/EHEP_ALICE_INDIA_School_2017/sample_codes/friend.C

this.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Line {
public:
  void setLength( double len = 0) {length = len ;}
  double getLength( void ) {return length;}
  Line() {cout << "Object is being created" << endl;}
  ~Line() {cout << "Object is being deleted" << endl;}

  bool Compare( Line *line){
    return this->getLength() > line->getLength();
  }

private:
  double length;
};

int main( ) {

  Line *line1 = new Line;
  Line *line2 = new Line;
  line1->setLength(6.0);
  line2->setLength(6.0);
  if(line1->Compare(line2))
    cout<< "line1 is longer than line2"<<endl;
  else
    cout<< "line1 is not longer than line2"<<endl;

  delete line1;
  delete line2;
  return true;
}
```

-:--- this.C All L2 (C++/l Abbrev)

(No changes need to be saved)

static.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>

using namespace std;

class Line {
public:
  static int objectCount ;
  static int getCounter(){return objectCount;}

  void setLength( double len = 0) {length = len ;}
  double getLength( void ) {return length;}

  Line() {cout << "Object is being created" << endl;
    objectCount++;}
  ~Line() {cout << "Object is being deleted" << endl;}

private:
  double length;
};

int Line::objectCount = 0;

int main( ) {
  Line line1;
  line1.setLength(6.0);

  Line line2(line1);
  Line line3;
  line3 = line2;

  cout << "Length of line1 : " << line1.getLength() <<endl;
  cout << "Length of line2 : " << line2.getLength() <<endl;
  cout << "Length of line3 : " << line3.getLength() <<endl;
  cout << "GetCounter : "<< Line::getCounter() <<endl;
  return true;
}
```

-:--- **static.C** All L2 (C++/l Abbrev)

(No changes need to be saved)

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

inherit.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Shape {
public:
  void setWidth(int w) { width = w;}
  void setHeight(int h) { height = h;}
protected:
  int width,height;
};

class PaintCost  {
public:
  int getCost(int area) { return area * 70;}
};

class Rectangle: public Shape, public PaintCost {
public:
  int getArea() { return (width * height); }
};

int main(void)
{
   Rectangle Rect;
   int area;

   Rect.setWidth(5);
   Rect.setHeight(7);
   area = Rect.getArea();

   cout << "Total area: " << Rect.getArea() << endl;
   cout << "Total paint cost: Rs." << Rect.getCost(area) << endl;

   return 0;
}
```

-:--- inherit.C All L3 (C++/l Abbrev)

inherit1.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Shape {
public:
  void setWidth(int w) { width = w;}
  void setHeight(int h) { height = h;}
private:
  int width,height;
};

class PaintCost {
public:
  int getCost(int area) { return area * 70;}
};

class Rectangle: public Shape, public PaintCost {
public:
  int getArea() { return (width * height); }
};

int main(void)
{
   Rectangle Rect;
   int area;

   Rect.setWidth(5);
   Rect.setHeight(7);
   area = Rect.getArea();

   cout << "Total area: " << Rect.getArea() << endl;
   cout << "Total paint cost: Rs." << Rect.getCost(area) << endl;

   return 0;
}
```

-:--- **inherit1.C** All L3 (C++/l Abbrev)

(No changes need to be saved)

inherit1.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Shape {
public:
  void setWidth(int w) { width = w;}
  void setHeight(int h) { height = h;}
  friend class Rectangle;

private:
  int width,height;
};

class PaintCost {
public:
  int getCost(int area) { return area * 70;}
};

class Rectangle: public Shape, public PaintCost {
public:
  int getArea() { return (width * height); }
};

int main(void)
{
   Rectangle Rect;
   int area;

   Rect.setWidth(5);
   Rect.setHeight(7);
   area = Rect.getArea();

   cout << "Total area: " << Rect.getArea() << endl;
   cout << "Total paint cost: Rs." << Rect.getCost(area) << endl;

   return 0;
}
```

-:--- **inherit1.C** All L9 (C++/l Abbrev)

Wrote /Data/EHEP_ALICE_INDIA_School_2017/sample_codes/inherit1.C

polymorphism.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Shape {
protected:
  int width, height;

public:
  Shape( int a = 0, int b = 0) : width(a), height(b){ }
  int area() { cout << "Parent class area :" <<endl; return 0; }
};

class Rectangle: public Shape {
public:
  Rectangle( int a = 0, int b = 0):Shape(a, b) { }
  int area () { cout << "Rectangle class area :" << (width * height) << endl ; return true;}
};

class Triangle: public Shape{
public:
  Triangle( int a = 0, int b = 0):Shape(a, b) { }
  int area () { cout << "Triangle class area :" << (width * height / 2) << endl; return true; }
};

int main( ) {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);

   shape = &rec;
   shape->area();

   shape = &tri;
   shape->area();

   return true;
}
```

-:--- **polymorphism.C** All L3 (C++/l Abbrev)

(No changes need to be saved)

virtual.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Shape {
protected:
  int width, height;

public:
  Shape( int a = 0, int b = 0) : width(a), height(b){ }
  virtual int area() { cout << "Parent class area :" <<endl; return 0; }
};

class Rectangle: public Shape {
public:
  Rectangle( int a = 0, int b = 0):Shape(a, b) { }
  int area () { cout << "Rectangle class area :" << (width * height) << endl ; return true;}
};

class Triangle: public Shape{
public:
  Triangle( int a = 0, int b = 0):Shape(a, b) { }
  int area () { cout << "Triangle class area :" << (width * height / 2) << endl; return true; }
};

int main( ) {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);

   shape = &rec;
   shape->area();

   shape = &tri;
   shape->area();

   return true;
}
```

-:--- virtual.C All L36 (C++/l Abbrev)

(No changes need to be saved)

purevirtual.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Shape {
protected:
  int width, height;

public:
  Shape( int a = 0, int b = 0) : width(a), height(b){ }
  virtual int area() = 0 ;
};

class Rectangle: public Shape {
public:
  Rectangle( int a = 0, int b = 0):Shape(a, b) { }
  int area () { cout << "Rectangle class area :" << (width * height) << endl ; return true;}
};

class Triangle: public Shape{
public:
  Triangle( int a = 0, int b = 0):Shape(a, b) { }
  int area () { cout << "Triangle class area :" << (width * height / 2) << endl; return true; }
};

int main( ) {
   Shape *shape;
   Rectangle rec(10,7);
   Triangle  tri(10,5);

   shape = &rec;
   shape->area();

   shape = &tri;
   shape->area();

   return true;
}
```

-:--- **purevirtual.C** All L3 (C++/l Abbrev)

(No changes need to be saved)

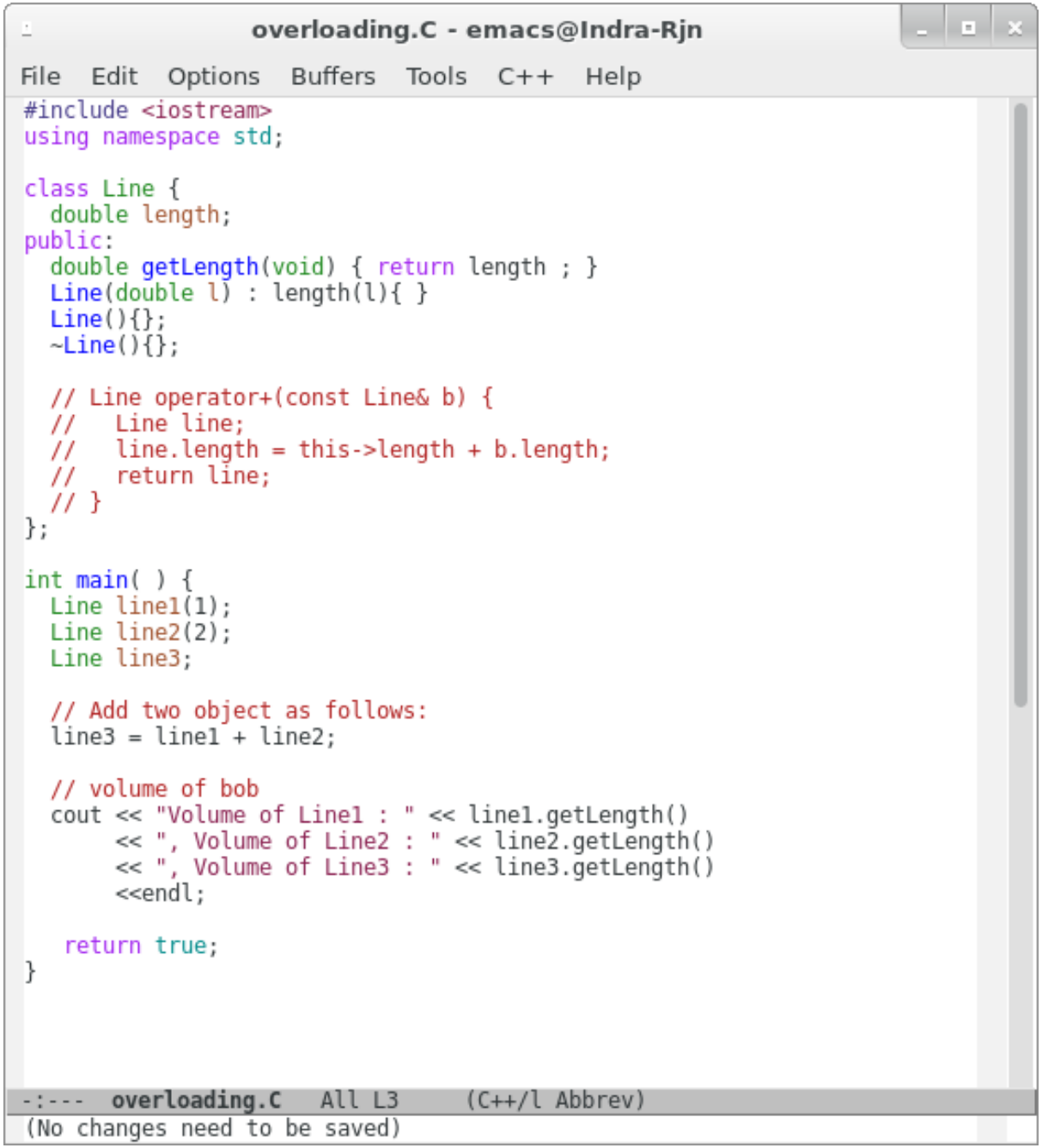
overloading.C - emacs@Indra-Rjn

File Edit Options Buffers Tools C++ Help

```
#include <iostream>
using namespace std;

class Line {
  double length;
public:
  double getLength(void) { return length ; }
  Line(double l) : length(l){ }
  Line(){};
  ~Line(){};

  Line operator+(const Line& b) {
    Line line;
    line.length = this->length + b.length;
    return line;
  }
};

int main( ) {
  Line line1(1);
  Line line2(2);
  Line line3;

  // Add two object as follows:
  line3 = line1 + line2;

  // volume of bob
  cout << "Volume of Line1 : " << line1.getLength()
       << ", Volume of Line2 : " << line2.getLength()
       << ", Volume of Line3 : " << line3.getLength()
       <<endl;

   return true;
}
```

-:--- **overloading.C** All L3 (C++/l Abbrev)

(No changes need to be saved)


overloading.C - emacs@Indra-Rjn
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;
class Line {
double length;
public:
double getLength(void) { return length ; }
Line(double l) : length(l){ }
Line(){};
~Line(){};
// Line operator+(const Line& b) {
// Line line;
// line.length = this->length + b.length;
// return line;
// }
};
int main( ) {
Line line1(1);
Line line2(2);
Line line3;
// Add two object as follows:
line3 = line1 + line2;
// volume of bob
cout << "Volume of Line1 : " << line1.getLength()
<< ", Volume of Line2 : " << line2.getLength()
<< ", Volume of Line3 : " << line3.getLength()
<<endl;
return true;
}
-:--- overloading.C All L3 (C++/l Abbrev)
(No changes need to be saved)

THANK YOU