DEPARTAMENTO DE FÍSICA NUCLEAR E ALTAS ENERGIAS

Computação e Linguagem de Programação

Aula 5 parte 2

Professores
Sandro Fonseca de Souza
Dilson de Jesus Damião

Sumário

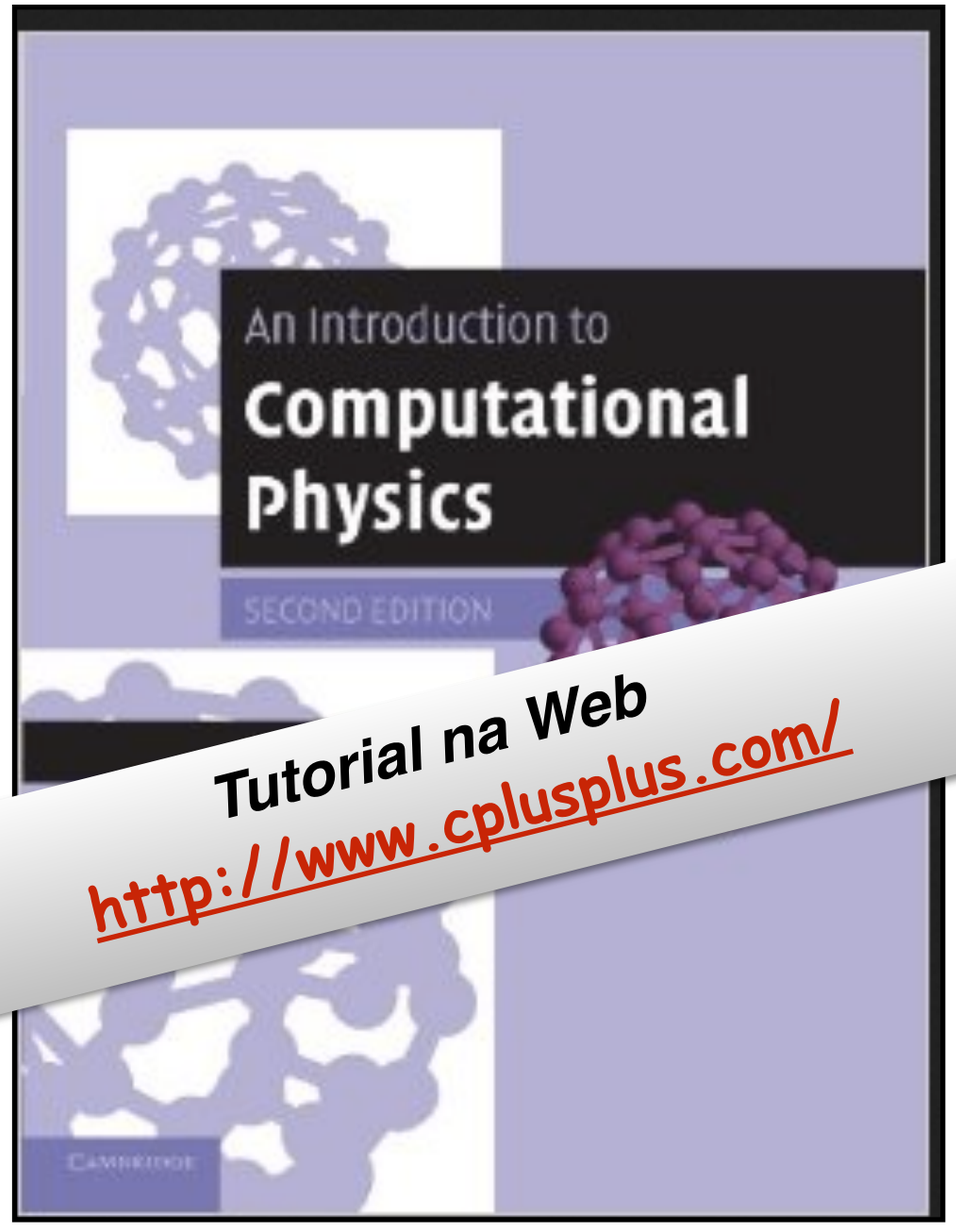
- **Linguagem de Programação C++**
  - **Tipos de dados;**
  - **Operadores;**
  - **Strings;**
  - **Funções intrínsecas**

Aula Anterior

- **Bibliografia Sugerida**
- **Motivações em FAE**
- **Introdução à Programação**
- **Linguagem de Programação C++ (parte 1)**

Bibliografia Sugerida

Bibliotecas padrão do C++
http://www.cplusplus.com/reference/



Tutorial na Web
http://www.cplusplus.com/

Data types

- Os data types determinam o tipo do dado que irão ser armazenados na memória do computador (RAM).

C++ provides 6 fundamental data types:

```
char
int
float
double
bool
wchar_t
```

There are also some qualifiers that can be put in front of the numerical data types to form derivatives:

```
short, long, signed, unsigned
```

For example:

```
short int
unsigned char
```

Data types

- A tabela mostra o fundamental dos data types em C++.

Group	Type names*	Notes on size / precision
Character types	**char**	Exactly one byte in size. At least 8 bits.
	char16_t	Not smaller than char. At least 16 bits.
	char32_t	Not smaller than char16_t. At least 32 bits.
	wchar_t	Can represent the largest supported character set.
Integer types (signed)	**signed char**	Same size as char. At least 8 bits.
	signed **short** *int*	Not smaller than char. At least 16 bits.
	signed **int**	Not smaller than short. At least 16 bits.
	signed **long** *int*	Not smaller than int. At least 32 bits.
	signed **long long** *int*	Not smaller than long. At least 64 bits.
Integer types (unsigned)	**unsigned char**	(same size as their signed counterparts)
	unsigned short *int*	
	unsigned *int*	
	unsigned long *int*	
	unsigned long long *int*	
Floating-point types	**float**	
	double	Precision not less than float
	long double	Precision not less than double
Boolean type	**bool**	
Void type	**void**	no storage
Null pointer	**decltype(nullptr)**	

*Certos tipos inteiros podem ser abreviados sem seus componentes - signed (sinal) ou int (inteiros) - somente a parte sem itálico é que necessita identificar o tipo.

Identificadores

- Um identificador é uma string de caracteres alfanuméricos que é usado para nomear variáveis, constantes, funções, estruturas ou classes.

Um identificador valido:

  - deve começar com uma letra ou underscore (_) ;
  - pode consistir somente por letras (a-z,A-Z), dígitos (0-9) e (_) ;
  - não deve usar qual palavra reservada para C++ que são:

```
asm, auto, bool, break, case, catch, char, class, const,
const_cast, continue, default, delete, do, double,
dynamic_cast, else, enum, explicit, export, extern, false, float,
for, friend, goto, if, inline, int, long, mutable, namespace, new,
operator, private, protected, public, register, reinterpret_cast,
return, short, signed, sizeof, static, static_cast, struct,
switch, template, this, throw, true, try, typedef, typeid,
typename, union, unsigned, using, virtual, void, volatile,
wchar_t, while
```

Identificadores

- Os seguintes identificadores são válidos:
  - mass
  - peynir
  - pos12
  - speed_of_light
  - SpeedOfLight
  - isPrim
- Os seguintes identificadores NÃO são válidos:
  - 2ndBit
  - speed of light
  - yağmur
  - c++
  - float

EM C++, faz distinção em identificadores com letras maiúsculas e minúsculas. (por exemplo:Casa e casa)

Variáveis

- **Exemplo de declarações:**

```
int i, j;
long k;
float w, x, y, z;
double speed, dragForce;
```

- **Quando a variável é declarada, você pode isso de duas formas distintas, mais equivalentes.**

```
int cake = 122;
```

```
int cake(122);
```

Variáveis

- **Exemplo de um programa com declarações:**

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

int main () {
  short x = 22, y = 11, z;
  z = x - y;
  cout << "z = " << z << endl;

  int p = 3;
  int q = x*y*z - 2*p;
  cout << "q = " << q << endl;

  return 0;
}
```

```
z = 11

q = 2656
```

Variáveis

- **Escopos aninhados e paralelos**

```
#include <iostream>
using namespace std;
int k = 11; // this k is global

int main ()
{
 int k = 22; // this k is local in main()
 {
  int k = 33; // this k is local in this block
  cout << "Inside internal block: k = " << k << endl;
 }
 cout << "Inside main(): k = " << k << endl;
 cout << "Global k = " << ::k << endl;

 return 0;
} // end main() block
```

```
Inside internal block: k = 33
Inside main(): k = 22
Global k = 11
```

Constantes

- **Para ajudar a promover a segurança , variáveis pode ser definida como constante usando o qualificador const. Elas não podem ser atribuídas durante a execução que deve ser inicializado no ponto de execução.**

```
const float PI = 3.1415926, TWOPI = 2.0*PI;

const int EOF = -1;
```

- **Os símbolos constantes, que não consomem memória (memory-consuming), são definidos via o # define.**

```
#define PI 3.1415926

#define MAX 100

#define NEWLINE '\n'
```

Constantes

- Às vezes queremos atribuir valores numéricos às palavras, por exemplo, Janeiro = 1, Fevereiro = 2, e assim por diante. O C ++ permite definir enumeração constante com *enum* palavra-chave.

```
enum { RED = 1, GREEN, BLUE };
```

é uma abreviação para

```
const int RED = 0, GREEN = 1, BLUE = 2;
```

- Enumeração é iniciada por padrão, com zero, mas podemos substituir esse por:

```
enum { RED = 1, GREEN = 3, BLUE = 7 };
```

- Se não for atribuído explicitamente, cada valor é maior do que anterior.

```
enum { RED = 1, GREEN, BLUE };
```

é equivalente a

```
enum { RED = 1, GREEN = 2, BLUE = 3 };
```

Constantes

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

int main ()
{
  short int m;
  enum {Jan=1, Feb, Mar, Apr, May,
        Jun, Aug, Sep, Oct, Nov, Dec};

  m = Apr;

  cout << "m =\t" << m << endl;
  cout << "Physics\nEngineer\n";
  cout << "Hello!\a" << endl;

  return 0;
}
```

```
m  =      4
Physics
Engineer
Hello!
```

Constantes

- Para strings literais, podemos usar aspas simples para um caractere, e aspas duplas para um um ou mais de um caractere.

```
'A'             // a single character
"B"             // a single character
"Hello World"   // a set of characters
```

- Há caracteres adicionais literais chamados códigos de escape ou sequências de escape que são precedidos por uma barra invertida (\).

Escape Code	Description	Example
\a	alert (beep)	cout << "Error !\a";
\n	newline	cout << "Gazi\nantep";
\t	horizontal tab	cout << x << '\t' << y;

✓Em C ++, existem apenas dois literais booleanos válidos true e false. Estes são expressos como valores de tipo booleano (bool).

Constantes

- Constantes literais de números inteiros podem ser representados por três bases diferentes: base-10 (decimal), base-8 (octal) e base-16 (hexadecimal)

```
i = 75;    // default base-10
i = 0113; // base-8
i = 0x4B; // base-16
i = 0x4b; // base-16
```

- Pontos flutuantes literais podem ser expressos com números decimais e / ou expoentes. O símbolo E ou E é utilizado como expoente.

```
x = 123.456;     // decimal real number
x = 1234.56e-1; // exponent (means 1234.56x10^-1)
c = 1.6E-19;     // exponent (means 1.6x10^-19)
A = 6.02e23;     // exponent (means 6.02x10^23)
```

Operadores básicos

Operadores são símbolos especiais que realizam operações sobre as variáveis e constantes.

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	13 + 5	18
-	Subtraction	13 - 5	8
*	Multiplication	13 * 5	65
/	Division	13 / 5	2
%	Modulus (reminder from x/y)	13 % 5	3

```
2 - 3 * 4 + 2 = -8
2 * 3 + 4 - 2 = 8
2 * (3 + 4) - 2 = 12
3 * 5 / 3 = 5
10 / 2 * 3 = 15
```

Assignment Operator (=)

```
int x, y;

x = 2;

y = 5*x;    // y = 10

x = x + 4; // x = 6

y = y/2;    // y = 5
```

chained assignment

```
m = (n = 66) + 9; // n = 66 and m = 75

x = y = 22;          // x = 22 and y = 22
```

Compound Assignment Operators (+=, -=, *=, /=, %=)

Operator	Description	Example	Equivalent to
+=	add and assign	x += 3	x = x + 3
-=	subtract and assign	x -= 5	x = x - 5
*=	multiply and assign	x *= 4	x = x * 4
/=	divide and assign	x /= 2	x = x / 2
%=	find reminder and and assign	x %= 9	x = x % 9

Operadores ++ e —

```
a = 5;

b = a++; // a = 6 and b = 5

c = ++a; // a = 7 and c = 7
```

```
x = x + 1;

x += 1;

x++;
```

Integer Division

```
int i, j, k;

double p, q;

i = 4/2;      // i = 2

j = 5/2;      // j = 2

p = 5/2;      // p = 2.0

p = 5/2.0;    // p = 2.5

q = i + p;    // q = 2.0 + 2.5 = 4.5;

k = 25.0/2;   // k = 12
```

Type Casting

```
int i; float f; double d;

i = int(7.25);    // i = 7

d = double(5);    // d = 5.0

f = float(7)/2;   // f = 3.5f
```

Strings Básicas

- Uma string é uma série de caracteres, como "Hello World!"
- Há três maneiras de definir uma string:

```
char *str1  = "This is string1";  // in C/C++
char str2[] = "This is string2";  // in C/C++
string str3 = "This is string3";  // in C++
```

- Algumas operações básicas pode ser feitas.

```
string s1, s2, s2, s4, s4;
s1 = "centi";
s2 = "meter";
s3 = s1;            // s3 = "centi" now
s4 = s1 + s2;       // s4 = "centimeter" now
s1 += "lmen";       // s1 = "centilmen" now
```

Strings Básicas

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

int main ()
{
  string name;

  cout << "What is your name? ";
  cin >> name;
  cout << "Hello " << name << endl;

  return 0;
}
```

```
What is your name? Mert

Hello Mert
```

Arquivos de cabeçalho

- #include permite que o programa para usar o código-fonte de outro arquivo.
- #include <iostream> refere-se a um arquivo externo chamado iostream, e diz ao processador para levar o arquivo iostream e inserir no programa atual.

Table 2.1: C++ standard library header files

C++ Standard Library	Standard Template Library	C Standard Library
ios	vector	cassert
iostream	deque	cctype
iomanip	list	cerrno
fstream	map	climits
sstream	set	clocale
	stack	cmath
	queue	csetjmp
	bitset	csignal
	algorithm	cstdarg
	functional	cstddef
	iterator	cstdio
		cstdint
		cstdlib
		cstring
		ctime

Funções Básicas Intrínsecas

Uma função de biblioteca é uma função fornecida pela linguagem C ++. Por exemplo, a biblioteca cmath que contém funções matemáticas/constantes:

Some C++ library mathematical functions and constants defined in <cmath>

Function Decleration	Description	Example	Result
`double fabs(double x);`	absolute value of real number, |x|	`fabs(-4.0)`	4.0
`int floor(double x);`	round down to an integer	`floor(-2.7)`	-3
`int ceil(double x);`	round up to an integer	`ceil(-2.7)`	-2
`double sqrt(double x);`	square root of x	`sqrt(4.0)`	2.0
`double pow(double x, double y);`	the value of x^y	`pow(2., 3.)`	8.0
`double exp(double x);`	the value of e^x	`exp(2.0)`	7.38906
`double log(double x);`	natural logarithm, $\log_e x = \ln x$	`log(4.0)`	1.386294
`double log10(double x);`	base 10 logarithm, $\log_{10} x = \log x$	`log10(4.0)`	0.602060
`double sin(double x);`	sinus of x (x is in radian)	`sin(3.14)`	0.001593
`double cos(double x);`	cosine of x (x is in radian)	`cos(3.14)`	-0.999999
`double tan(double x);`	tangent of x (x is in radian)	`tan(3.14)`	-0.001593
`double asin(double x);`	arc-sine of x in the range [-pi/2, pi/2]	`asin(0.5)`	0.523599
`double acos(double x);`	arc-cosine of x in the range [-pi/2, pi/2]	`acos(0.5)`	1.047198
`double atan(double x);`	arc-tangent of x in the range [-pi/2, pi/2]	`atan(0.5)`	0.463648
`M_PI`	constant pi	`myPI = M_PI`	3.141592...
`M_E`	constant e	`x = M_E`	2.718281...

Funções Básicas Intrínsecas

Some standard C++ library functions and constant defined in <cstdlib>

Function Decleration	Description	Example	Result
`int abs(int x);`	absolute value of integer number, |x|	`abs(-4)`	4
`int atoi(const char *s);`	converts string to integer	`atoi("-1234")`	-1234
`double atof(const char *s);`	converts a string to double	`atof("123.54")`	123.54
`void exit(int status);`	terminates the calling process "immediately"	`exit(1)`	-
`int rand(void);`	Returns a random integer between 0 and RAND_MAX	`rand()`	1048513214
RAND_MAX	The largest number rand() will return	`x = RAND_MAX`	2147483647

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

int main ()
{
  double beta;
  cout << "Input an angle in degrees: ";
  cin >> beta;

  // convert from degrees to radians
  beta = beta * M_PI/180.0;
  cout << "sin(beta) = " << sin(beta) << endl;
  cout << "cos(beta) = " << cos(beta) << endl;
  cout << "tan(beta) = " << tan(beta) << endl;

  return 0;
}
```

```
Input an angle in degrees: 60
sin(beta) = 0.866025
cos(beta) = 0.5
tan(beta) = 1.73205
```

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

int main (){
  double x;
  cout << "a value ";
  cin >> x;

  cout << "log(x)     = " << log(x)      << endl;
  cout << "log10(x)   = " << log10(x)    << endl;
  cout << "exp(x)     = " << exp(x)      << endl;,
  cout << "pow(x,2.5)= " << pow(x,2.5) << endl;
  return 0;
}
```

```
a value 1.4
log(x)     = 0.336472
log10(x)   = 0.146128
exp(x)     = 4.0552
pow(x,2.5)= 2.3191
```

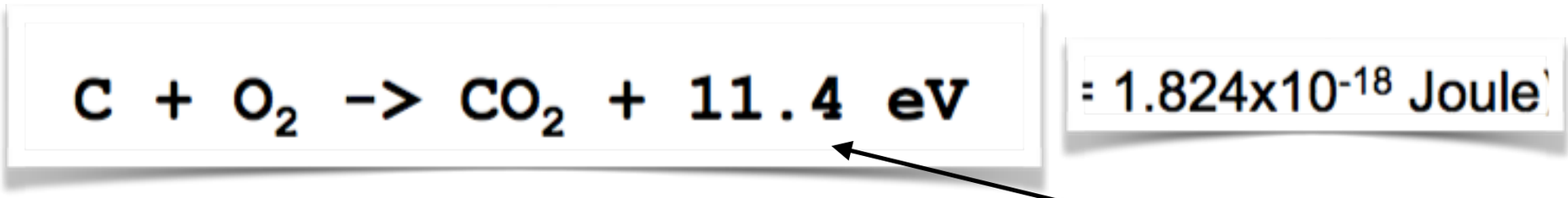
Exemplos

$N_A = 6.022x10^{23}$ atoms/mole

$M_C = 12$ g/mole

Exemplo 1

Os motores a gasolina usar o calor produzido na combustão do carbono e hidrogénio na gasolina. Uma das mais importantes fontes de energia é a de oxidação do carbono para formar o dióxido de carbono



Escrever um programa para encontrar o número total de átomos de carbono e a energia total libertada quando m (kg) de carbono é oxidado em que m é a entrada a partir do teclado.

Exemplo 1

Solução

```
Input the mass of the carbon in kg: 1
Number of C atoms   = 5.01833e+25
Total energy in J   = 9.15344e+07
```

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

int main (){
  const double NA = 6.022e23;
  const double Energy_Per_Reaction = 1.824e-18, MC = 12.0;
  double m, nC, en;

  cout << "Input the mass of the carbon in kg: ";
  cin >> m;

  // Number of carbon atoms in m kg
  nC = 1000*m * NA / MC;

  // Total energy released in J
  en = nC * Energy_Per_Reaction;

  cout << "Number of C atoms   = " << nC << endl;
  cout << "Total energy in J   = " << en << endl;

  return 0;
}
```

Exemplo 2

Calcular o intervalo e tempo de voo de um projétil dada a v_0 velocidade inicial e ângulo de θ elevação.

Alcance

$$R = v_0^2 \sin(2\theta) \ / \ g$$

Tempo de vôo

$$T = v_0^2 \sin^2(\theta) \ / \ 2g$$

Exemplo 2

Solução

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

int main(){
  const double g = 9.81;
  double v0, theta, R, T;

  // get the values
  cout << "Input the speed (in m/s): ";
  cin  >> v0;
  cout << "Input the angle of elevation (in degrees): ";
  cin  >> theta;

  // convert angle into radian
  theta = theta * M_PI/180.0;

  // calculate R and T
  R = v0*v0 * sin(2.0*theta)/g;
  T = pow(v0*sin(theta),2.0) / (2*g);
  cout << "Projectile range = " << R << " m." << endl;
  cout << "Time of flight   = " << T << " s." << endl;
}
```

Exercícios

Computação e Linguagem de Programação

Exercícios

1.Quantos são os tipos de dados em C ++?

2.Qual é a diferença entre o short int e int (apresente exemplos)?

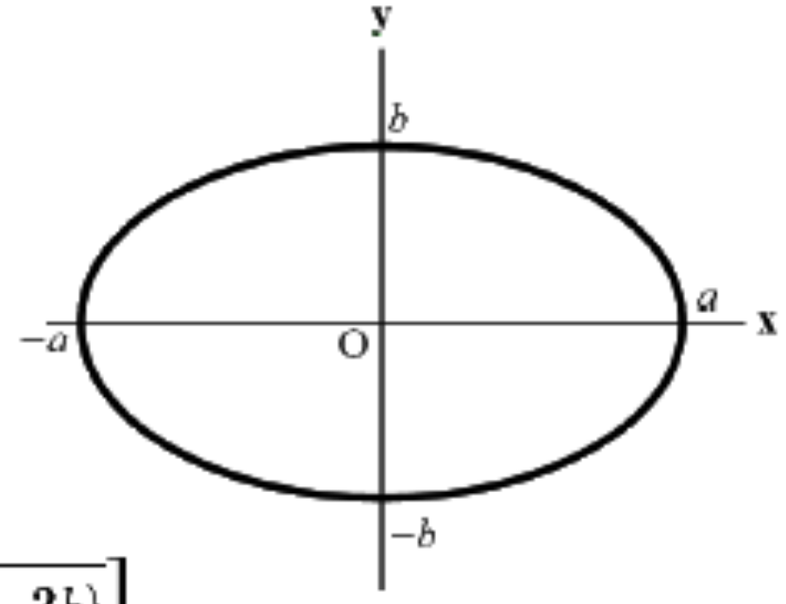
3.Qual a diferença entre double e float (apresente exemplos)?

4. Quantas formas existem para definir constantes em C++?

Exercícios

Exercício 1: A figura mostra uma elipse cujo eixo semi-principal é o comprimento de a, semi-eixo menor é o comprimento b. Escreva um programa C ++ que introduz os valores de a e b, e saídas de área (A) e na circunferência (C) de da elipse.

$$A = \pi ab$$

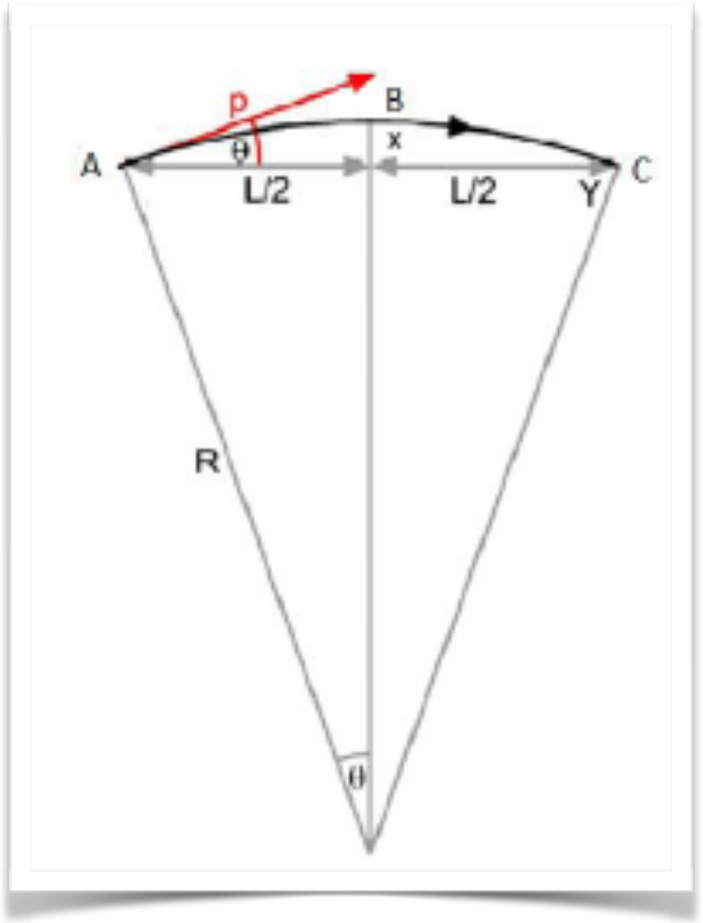


$$C \approx \pi \left[3(a+b) - \sqrt{(3a+b)(a+3b)} \right]$$

Exercícios

Exercício 2: Considere que uma fonte radioativaque emite partículas alfa com o momentum de p. Eles entram (no ponto A) para uma região contendo campo magnético uniforme B = 1,5 T (para fora da página) como mostrado na figura. As partículas segue o arco ABC. Ao medir Sagita (x é distância L) pode-se calcular o raio de curvatura do arco e, portanto, o momentum das partículas.

Escrever um programa em C ++ para a entrada x e L e de saída o raio de curvatura (R), em centímetros e momentum (**p**) em MeV/c das partículas alfa. Use cinemática relativística e a ordem de x (e de L) é em cm.



https://en.wikipedia.org/wiki/Sagitta_(geometry)

Próxima Aula

- **Operadores relacionais e lógicos**
- **Expressões boleadas**
- **Estrutura if**
- **Estrutura if .. else**
- **Estrutura if .. else if .. else**
- **Loop while**
- **Loop do.....while**
- **Loop for**
- **break e continue**
- **loops infinitos**
- **loops aninhados**
- **problemas resolvidos**