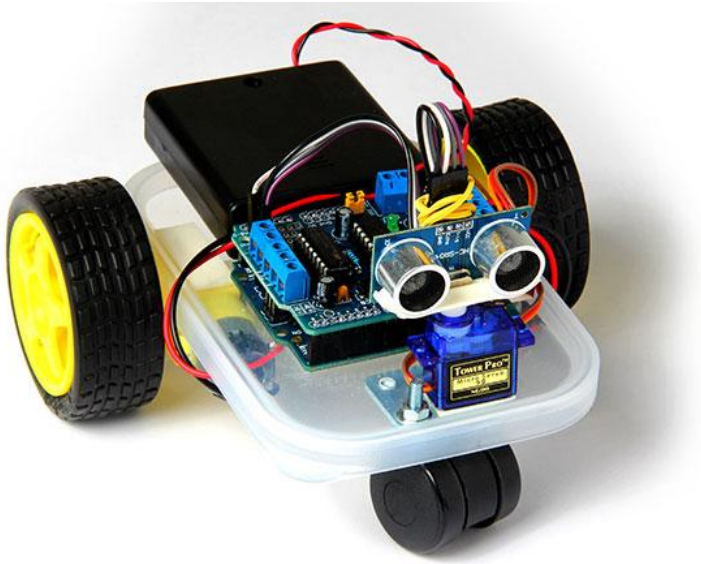
EP486 Microcontroller Applications

Topic 1

Review of C/C++



Department of Engineering Physics

University of Gaziantep

Sep 2013

Data Types

***C++ provides 6 fundamental data types*:**

char

int

float

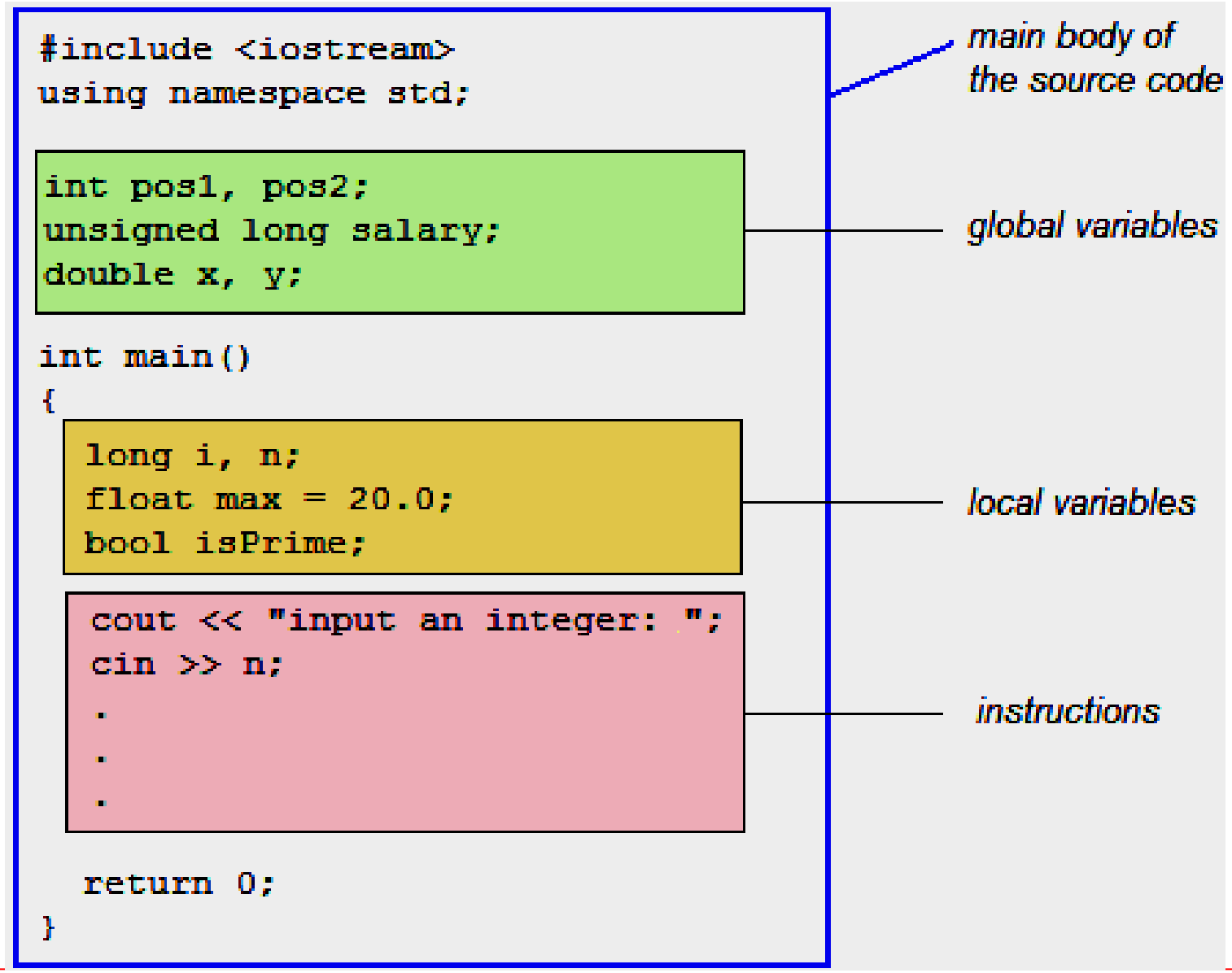
double

bool

wchar_t

```
int i, j = 0;
long k;
float w, x, y, z = 0.7;
double speed, dragForce;
```

In C++, a variable can have either *local* or *global* scope.



- Floating point literals express numbers with decimals and/or exponents. The symbol **E** or **e** is used in the exponent.

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

Basic Intrinsic Functions

An *intrinsic* or a *library* function is a function provided by C++ language.

For example the **`cmath`** library contains mathematical functions/constants:

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...

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

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

printf() Function

```
#include <cstdio>
```

Example	Output
`printf("Hello World\n");`	`Hello World`
`int x = 112;` `printf("x = %d\n");`	`x = 112`
`float y = 45.3;` `printf("y = %f\n");`	`y = 45.30000`

Type Specifiers

Tablo 4.2: *Tip karakterleri*

Tip Karakteri	Anlamı	Yazdırılacak veri tipi
`%c`	tek bir karakter	`char`
`%s`	karakter dizisi (string)	`char`
`%d`	işaretli ondalık tamsayı	`int, short`
`%ld`	uzun işaretli ondalık tamsayı	`long`
`%u`	işaretsiz ondalık tamsayı	`unsigned int, unsigned short`
`%lu`	işaretsiz uzun tamsayı	`unsigned long`
`%f`	Gerçel sayı	`float`
`%lf`	Çift duayarlı gerçel sayı	`double`

Relational Operators

Operator	Description	Example
<	less than	x < y
<=	less than or equal to	x <= y
>	greater than	x > y
>=	greater than or equal to	x >= y
==	equal to	x == y
!=	not equal to	x != y

Logical Operators

Operator	Description	Example
&&	logical AND, conjuction. Both sides must be true for the result to be true	x > 2 && y == 3
||	Logical OR, disjunction. The result is true if either side or both sides are true.	x > 2 || x <= 9
!	Logical NOT, negation	! (x>0)

if … else structure

```
// Quadratic roots
#include <iostream>
#include <cmath>
using namespace std;

int main() {
  double a, b, c;
  cin >> a >> b >> c;

  double Delta = b*b - 4*a*c;

  if ( Delta < 0. ) {
    cout << "The roots are imaginary!" << endl;

  } else if ( Delta == 0. ) {
    double x1 = -b / (2*a);
    cout << "The root is " << x1 << endl;

  } else {
    double x1 = ( -b - sqrt(Delta) ) / (2*a);
    double x2 = ( -b + sqrt(Delta) ) / (2*a);
    cout << "The two roots are "<< x1<< " and " << x2<< endl;

  }
}
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Loops

Calculate the series sum: 1 + 1/2 + 1/3 + 1/4 + 1/5 + ... + 1/n.

```
int main() {

  cout << "Input n: ";
  int n;
  cin >> n;

  float k=1, s=0;


  while (k<=n) {
    s = s + 1.0/k;
    k++;
  }

  cout << s << endl;
}
```

```
int main() {

  cout << "Input n: ";
  int n;
  cin >> n;

  float k=1, s=0;


  do{
    s = s + 1.0/k;
    k++;
  }while(k<=n);

  cout << s << endl;
}
```

Calculate the series sum: 1 + 1/2 + 1/3 + 1/4 + 1/5 + ... + 1/n.

```
int main() {

  cout << "Input n: ";
  int n;
  cin >> n;

  float k=1, s=0;


  for(k=1; k<=n; k++){
    s = s + 1.0/k;
  }

  cout << s << endl;
}
```

Jump Statements: `break` and `continue`

```
// break statement

int main()
{
  double x;

  for(int i =-3; i<=3; i++)
  {
     if(i==0) break;
     x = 1.0/i;
     cout << x << endl;
  }
}
```

```
-0.3333
-0.5
-1
```

```
// continue statement

int main()
{
  double x;

  for(int i =-3; i<=3; i++)
  {
     if(i==0) continue;
     x = 1.0/i;
     cout << x << endl;
  }
}
```

```
-0.3333
-0.5
-1
1
0.5
0.3333
```

Arrays

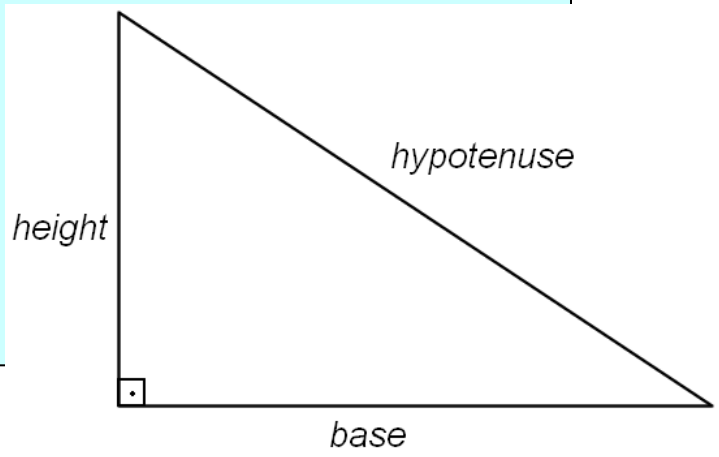
```
int main()
{
   const int n = 5;

   float  a[ ] = {8.4, 3.6, 9.1, 4.7, 3.9};
   int    b[n] = {4, 2};
   double c[n] = {0.0};


   for (int i = 0; i<n; i++)
     cout << a[i] << b[i] << c[i] << endl;

}
```

Functions

```
// the Function
double factorial(int n)
{
    double f = 1.0;
    for(int i=1; i<=n; i++){
        f *= i;
    }
    return f;
}

int main(){
  cout << factorial(10) << endl;
}
```

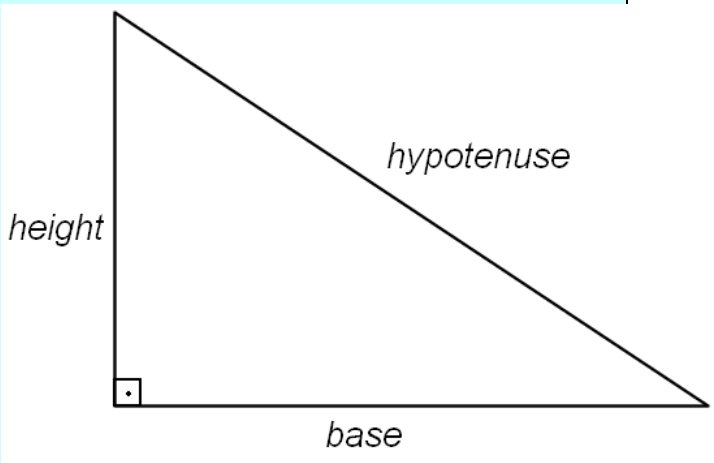
```
#include <iostream>
Using namespace std;

int x;

void setup(){
    x = 3;
}
void printDouble()
{
    cout << 2*x << endl;
}

int main(){
    setup();
    printDouble();
}
```

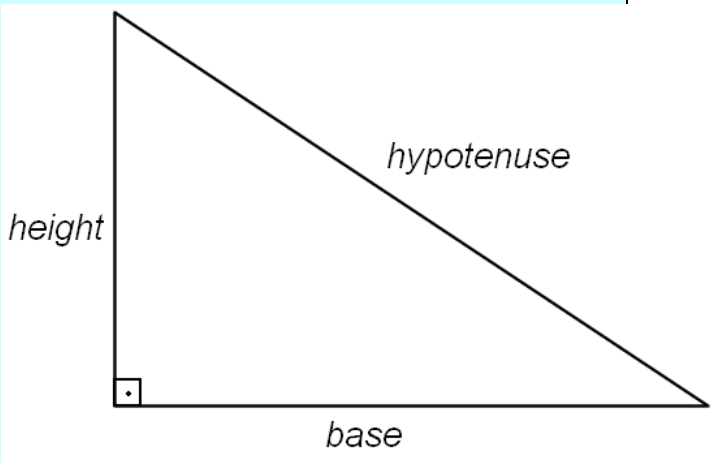
```
// Function 1
double area(double base, double height){
    return base * height / 2.0;
}

// Function 2
double hypo(double base, double height){
    return sqrt(base*base + height*height);
}

int main(){
  double b = 3.0 ,h = 4.0;

  cout << area(b, h) << endl;
  cout << hypo(b, h) << endl;
}
```



Classes - 1

```
class Triangle{
 public:
    double base, height;
    Triangle(double B, double H){
       base = B; height=H;
    }
    double area(){
       return base * height / 2.0;
    }
    double hypo(){
     return sqrt(base*base + height*height);
    }
};

int main(){
  Triangle T(3,4);

  cout << T.area() << endl;
  cout << T.hypo() << endl;
  cout << T.base   << endl;
}
```



Classes - 2

```
class Triangle{
 public:
    double base, height;
    Triangle(double B, double H){
       base = B; height=H;
    }
    double area(){
       return base * height / 2.0;
    }
    double hypo(){
     return sqrt(base*base + height*height);
    }
};

int main(){
  Triangle* T = new Triangle(3,4);

  cout << T->area() << endl;
  cout << T->hypo() << endl;
  cout << T->base   << endl;
}
```