Exam Practice!

1. T
2. F
3. T
4. T
5. F
6. Not Correct. int i, j, k;
7. Correct.
8. Incorrect. double D1, D2, D3;
9. Correct.
10. Correct.
11. (d)
12. (b)
13. (a)
14. (c)
15. (e)

Memory Snapshots

16. x1=>2, z=>2, x=>2
17. x=>2, y=>1, a=>3.8, n=>2

Output

18. value_1 = 5.78263
19. Missing ; (value_4 = 6.645832e+01)
20. value_5 = 7750

Programming Exercises

```
/*-------------------------------------------------------------------*/
/*  Problem chapter2_21                                              */
/*                                                                   */
/*  This program converts miles to kilometers.                       */

#include <iostream>
using namespace std;

int main()
{
```

```
   /*  Declare variables.  */
   double miles, kilometers;

   /*  Enter number of miles from the keyboard.  */
   cout << "Enter the number of miles: \n";
   cin >> miles;

   /*  Compute the number of kilometers equal to the specified miles. */
   kilometers = 1.6093440*miles;

   /*  Print the number of kilometers.  */
   cout << miles << " miles = " << kilometers << " kilometers \n";

   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------*/
/*  Problem chapter2_22                                                     */
/*                                                                          */
/*  This program converts meters to miles.                                  */
```

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

int main()
{
   /*  Declare variables.  */
   double miles, meters;

   /*  Enter number of meters from the keyboard.  */
   cout << "Enter the number of meters: \n";
   cin >> meters;

   /*  Compute the number of miles equal to the specified meters. */
   miles = meters/1609.3440;

   /*  Print the number of miles.  */
   cout  << meters << " meters = "<< miles << " miles \n";

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_23                                                    */
/*                                                                         */
/*  This program converts pounds to kilograms.                             */

#include <iostream>
using namespace std;

int main()
{
   /*  Declare variables.  */
   double pounds, kilograms;

   /*  Enter number of pounds from the keyboard.  */
   cout << "Enter the number of pounds: ";
   cin >> pounds;

   /*  Compute number of kilograms equal to the specified pounds.  */
   kilograms = pounds/2.205;

   /*  Print the number of kilograms.  */
   cout << pounds << " pounds = " << kilograms << " kilograms \n";

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_24                                                    */
/*                                                                         */
/*  This program converts newtons to pounds.                               */

#include <iostream>
```

```
using namespace std;
int main()
{
   /*  Declare variables.  */
   double pounds, newtons;

   /*  Enter number of newtons from the keyboard.  */
   cout << "Enter the number of newtons:  ";
   cin >> newtons;

   /*  Compute number of pounds equal to the specified newtons.  */
   pounds = newtons/4.448;

   /*  Print the number of pounds.  */
   cout << newtons << " newtons = " << pounds << " pounds \n";

   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------*/

/*--------------------------------------------------------------------*/
/*  Problem chapter2_25                                               */
/*                                                                    */
/*  This program converts degrees Fahrenheit to degrees Rankin.       */

#include <iostream>
using namespace std;

int main()
{
   /*  Declare variables.  */
   double degrees_F, degrees_R;

   /*  Enter temperture in degrees Fahrenheit from the keyboard.  */
   cout << "Enter the temperature in degrees Fahrenheit:  ";
   cin >> degrees_F;

   /*  Compute the equivalent temperature in degrees Rankin  */
   /*  from the given temperature.                           */
   degrees_R = degrees_F + 459.67;

   /*  Print the temperatures.  */
   cout << degrees_F << " degrees Fahrenheit = " << degrees_R << " degrees
Rankin \n";

   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------*/

/*--------------------------------------------------------------------*/
/*  Problem chapter2_26                                               */
/*                                                                    */
/*  This program converts degrees Celsius to degrees Rankin.          */

#include <iostream>
```

```
using namespace std;

int main()
{
   /*  Declare variables.  */
   double degrees_C, degrees_F, degrees_R;

   /*  Enter temperture in degrees Celsius from the keyboard.  */
   cout << "Enter the temperature in degrees Celsius:  \n";
   cin >> degrees_C;

   /*  Compute the equivalent temperature in degrees Rankin  */
   /*  from the given temperature.                           */
   degrees_F = (9.0/5.0)*degrees_C + 32;
   degrees_R = degrees_F + 459.67;

   /*  Print the temperatures.  */
   cout << degrees_C << " degrees Celsius = " << degrees_R << " degrees
Rankin \n";

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_27                                                    */
/*                                                                         */
/*  This program converts degrees Kelvin to degrees Fahrenheit.            */

#include <iostream>
using namespace std;

int main()
{
   /*  Declare variables.  */
   double degrees_R, degrees_K, degrees_F;

   /*  Enter temperture in degrees Kelvin from the keyboard.  */
   cout << "Enter the temperature in degrees Kelvin:  \n";
   cin >> degrees_K;

   /*  Compute the equivalent temperature in degrees Fahrenheit  */
   /*  from the given temperature.                               */
   degrees_R = (9.0/5.0)*degrees_K;
   degrees_F = degrees_R - 459.67;

   /*  Print the temperatures.  */
   cout << degrees_K << " degrees Kelvin = " << degrees_F << " degrees
Fahrenheit \n";

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
```

```
/*  Problem chapter2_28                                                 */
/*                                                                      */
/*  This program finds the area of a  rectangle.                        */

#include <iostream>
using namespace std;

int main()
{
   /*  Declare variables.  */
   double a, b, area;

   /*  Enter the lengths of sides of the rectangle.  */
   cout << "Enter the lengths of the sides of the rectangle:  ";
   cin >> a >> b;

   /*  Compute the area of the rectangle.  */
   area = a*b;

   /*  Print the value of the area.  */
   cout << "The area of a rectangle with sides " << a << " and " << b
        << " is " << area << endl;

   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------*/

/*--------------------------------------------------------------------*/
/*  Problem chapter2_29                                               */
/*                                                                    */
/*  This program finds the area of a triangle.                        */

#include <iostream>
using namespace std;

int main()
{
   /*  Declare variables.  */
   double h, b, area;

   /*  Enter the base and the height of the triangle.  */
   cout << "Enter the base and the height of the triangle: ";
   cin >> b >> h;

   /*  Compute the area of the triangle.  */
   area = 0.5*b*h;

   /*  Print the value of the area.  */
   cout << "The area of a triangle with base " << b << " and height " << h
        << "is " << area << endl;

   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------------------*/
/*  Problem chapter2_30                                                    */
/*                                                                         */
/*  This program finds the area of a circle.                               */

#include <iostream>
using namespace std;

const double PI = 3.141593;

int main()
{
   /*  Declare variables.  */
   double r, area;

   /*  Enter the radius.  */
   cout << "Enter the radius of the circle: ";
   cin >> r;

   /*  Compute the area of the circle.  */
   area = PI*r*r;

   /*  Print the value of the area.  */
   cout << "The area of a circle with radius " << r << " is "
        << area << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_31                                                    */
/*                                                                         */
/*  This program computes the area of a sector of a circle when            */
/*  theta (u) is the angle in radians between the radii.                   */

#include <iostream>

using namespace std;

int main()
{
   /*  Declare variables.  */
   double u, r, area;

   /*  Enter the lengths of the radii and */
   /*  the angle between them.            */
   cout << "Enter the length of the radii and the angle "
        << "(in radians) between them: ";
   cin >> r >> u;

   /*  Compute the area of the sector.  */
   area = (r*r*u)/2.0;

   /*  Print the value of the area.  */
   cout << "The area of sector is " << area << endl;
```

```
   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------*/
/*  Problem chapter2_32                                                     */
/*                                                                          */
/*  This program computes the area of a sector of a circle when             */
/*  the input (d) is the angle in degrees between the radii.                */

#include <iostream>

using namespace std;

const double PI = 3.141593;

int main()
{
   /*  Declare variables.  */
   double d, r, area, theta;

   /*  Enter the lengths of the radii and */
   /*  the angle between them.            */
   cout << "Enter the length of the radii and the angle "
        << "(in degrees) between them: ";
   cin >> r >> d;

   /*  Compute the value of the angle in radians.  */
   theta = d * PI / 180;

   /*  Compute the area of the sector.  */
   area = (r*r*theta)/2.0;

   /*  Print the value of the area.  */
   cout << "The area of sector is " << area << endl;

   /*  Exit program.  */
   return 0;
}
/*--------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------*/
/*  Problem chapter2_33                                                     */
/*                                                                          */
/*  This program computes the area of an                                    */
/*  ellipse with semiaxes a and b.                                          */

#include <iostream>
using namespace std;

const double PI = 3.141593;

int main()
{
   /*  Declare variables.  */
```

```
   double a, b, area;

   /*  Enter the length of the semiaxes.  */
   cout << "Enter the length of the semiaxes: ";
   cin >> a >> b;

   /*  Compute the area of the ellipse.  */
   area = PI*a*b;

   /*  Print the value of the area.  */
   cout << "The area of an ellipse with semiaxes " << a << " and "
        << b << " is " << area << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_34                                                    */
/*                                                                         */
/*  This program computes the area of the surface                          */
/*  of a sphere of radius r.                                               */

#include <iostream>
using namespace std;

const double PI = 3.141593;

int main()
{
   /*  Declare variables.  */
   double r, area;

   /*  Enter the radius of the sphere.  */
   cout << "Enter the radius of the sphere: ";
   cin >> r;

   /*  Compute the area of the sphere.  */
   area = 4.0*PI*r*r;

   /*  Print the value of the area.  */
   cout << "The area of a sphere with radius " << r
        << " is " << area << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_35                                                    */
/*                                                                         */
/*  This program computes the volume                                       */
/*  of a sphere of radius r.                                               */

#include <iostream>
```

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

const double PI = 3.141593;

int main()
{
   /*  Declare variables.  */
   double r, volume;

   /*  Enter the radius of the sphere.  */
   cout << "Enter the radius of the sphere: ";
   cin >> r;

   /*  Compute the volume of the sphere.  */
   volume = (4.0/3)*PI*pow(r,3);

   /*  Print the value of the volume.  */
   cout << "The volume of a sphere with radius " << r
        << " is " << volume << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_36                                                    */
/*                                                                         */
/*  This program computes the volume of a cylinder                         */
/*  of radius r and height h.                                              */

#include <iostream>
using namespace std;

const double PI = 3.141593;

int main()
{
   /*  Declare variables.  */
   double r, h, volume;

   /*  Enter the radius and height of the cylinder.  */
   cout << "Enter the radius and the height of the cylinder: ";
   cin >> r >> h;

   /*  Compute the volume of the cylinder.  */
   volume = PI*r*r*h;

   /*  Print the volume.  */
   cout << "The volume of a cylinder of radius " << r << " and "
        << "height " << h << " is " << volume << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/
```

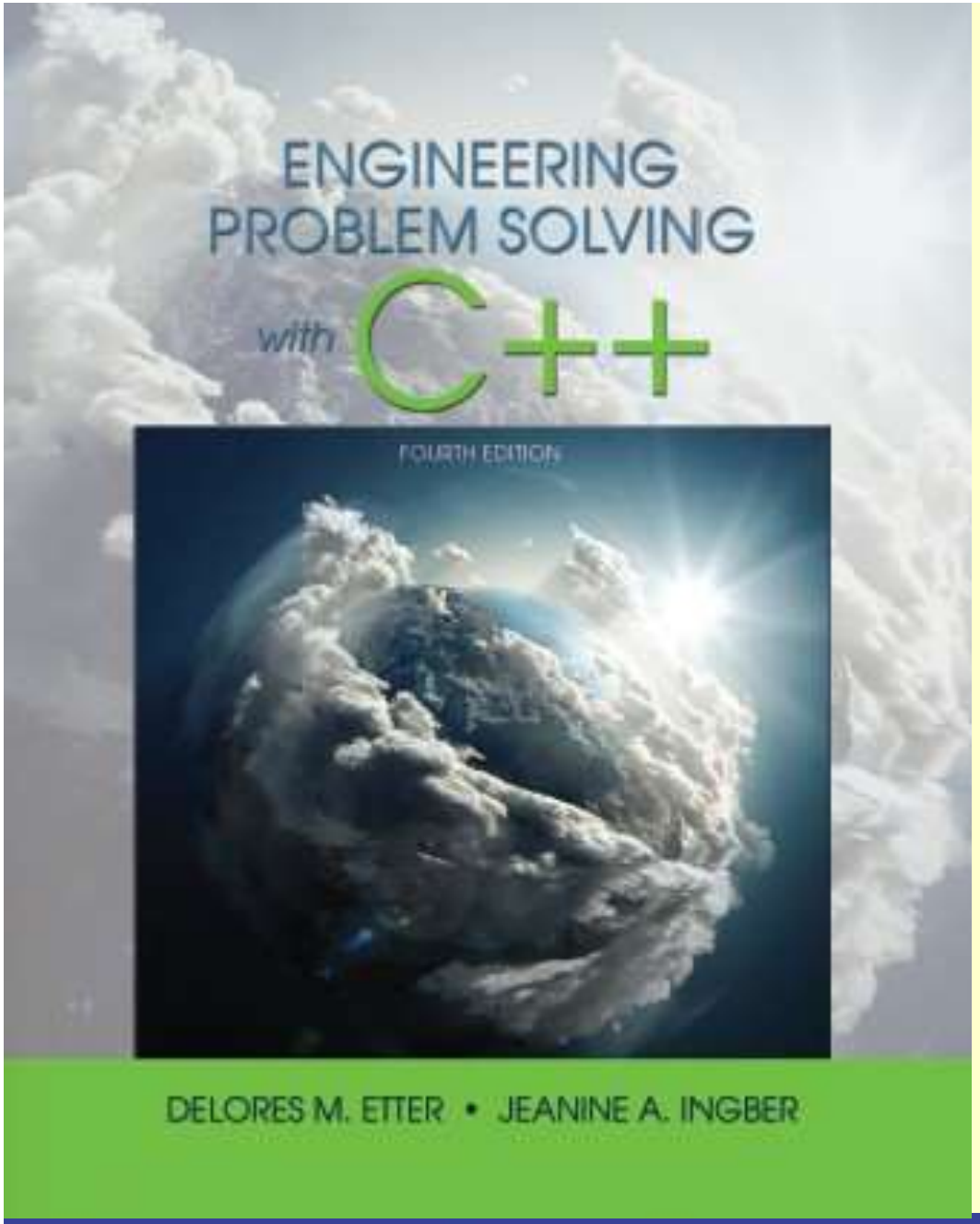
```
/*-------------------------------------------------------------------------*/
/*  Problem chapter2_37                                                    */
/*                                                                         */
/*  This program computes the molecular weight of the                      */
/*  amino acid glycine.                                                    */

#include <iostream>

using namespace std;

/*  Defines symbolic constants for the appropriate atomic weights.  */
const double OXYGEN = 15.9994;
const double CARBON = 12.011;
const double NITROGEN = 14.00674;
const double HYDROGEN = 1.00794;

int main()
{
   /*  Declare variables.  */
   double molecular_weight;

   /*  Compute the molecular weight of glycine.  */
   molecular_weight = (2*OXYGEN) + (2*CARBON) +
                      NITROGEN + (5*HYDROGEN);

   /*  Print the molecular weight.  */
   cout << "The molecular weight of glycine is " << molecular_weight << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_38                                                    */
/*                                                                         */
/*  This program computes the molecular weights of the                     */
/*  amino acids glutamic and glutamine.                                    */

#include <iostream>

using namespace std;

/*  Defines symbolic constants for the appropriate atomic weights.  */
const double OXYGEN = 15.9994;
const double CARBON = 12.011;
const double NITROGEN = 14.00674;
const double HYDROGEN = 1.00794;

int main()
{
   /*  Declare variables.  */
   double mol_weight_glutamic, mol_weight_glutamine;

   /*  Compute the molecular weights.  */
   mol_weight_glutamic = (4*OXYGEN) + (5*CARBON) +
```

```
                        NITROGEN + (8*HYDROGEN);

   mol_weight_glutamine = (3*OXYGEN) + (5*CARBON) +
                        (2*NITROGEN) + (10*HYDROGEN);

   /*  Print the molecular weights.  */
   cout << "The molecular weight of glutamic is " << mol_weight_glutamic
        << endl;
   cout << "The molecular weight of glutamine is " << mol_weight_glutamine
        << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_39                                                    */
/*                                                                         */
/*  This program computes the molecular weight of a particular             */
/*  amino acid given the number of atoms for each of the five              */
/*  elements found in the amino acid.                                      */

#include <iostream>
using namespace std;

/*  Defines symbolic constants for the appropriate atomic weights.  */
const double OXYGEN = 15.9994;
const double CARBON = 12.011;
const double NITROGEN = 14.00674;
const double HYDROGEN = 1.00794;
const double SULFUR = 32.066;

int main()
{
   /*  Declare variable.  */
   int no_oxy, no_carbon, no_nitro, no_hydro, no_sulfur;
   double molecular_weight;

   /*  Enter the number of atoms for each of the five elements.  */
   cout << "Enter the number of oxygen atoms found "
          "in the amino acid. \n";
   cin >> no_oxy;
   cout << "Enter the number of carbon atoms. \n";
   cin >> no_carbon;
   cout << "Enter the number of nitrogen atoms. \n";
   cin >> no_nitro;
   cout << "Enter the number of sulfur atoms. \n";
   cin >> no_sulfur;
   cout << "Enter the number of hydrogen atoms. \n";
   cin >> no_hydro;

   /*  Compute the molecular weight.  */
   molecular_weight = (no_oxy*OXYGEN) + (no_carbon*CARBON) +
                           (no_nitro*NITROGEN) + (no_sulfur*SULFUR) +
                           (no_hydro*HYDROGEN);
```

```
   /*  Print the molecular weight.  */
   cout << "The molecular weight of this particular amino acid is "
        << molecular_weight << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_40                                                    */
/*                                                                         */
/*  This program computes the average atomic weight of the atoms           */
/*  found in a particular amino acid given the number of atoms for         */
/*  each of the five elements found in amino acid.                         */

#include <iostream>
using namespace std;

/*  Defines symbolic constants for the appropriate atomic weights.  */
const double OXYGEN = 15.9994;
const double CARBON = 12.011;
const double NITROGEN = 14.00674;
const double HYDROGEN = 1.00794;
const double SULFUR = 32.066;

int main()
{
   /*  Declare variables.  */
   int no_oxy, no_carbon, no_nitro, no_hydro, no_sulfur, total_no;
   double average_atomic_weight;

   /*  Enter the number of atoms for each of the five elements.  */
   cout << "Enter the number of oxygen atoms found "
        <<  " in the amino acid. \n";
   cin >> no_oxy;
   cout << "Enter the number of carbon atoms. \n";
   cin >> no_carbon;
   cout << "Enter the number of nitrogen atoms. \n";
   cin >> no_nitro;
   cout << "Enter the number of sulfur atoms. \n";
   cin >> no_sulfur;
   cout << "Enter the number of hydrogen atoms. \n";
   cin >> no_hydro;

   /*  Compute the average weight of the atoms.  */
   total_no = no_oxy + no_carbon + no_nitro + no_sulfur + no_hydro;
   average_atomic_weight = ((no_oxy*OXYGEN) + (no_carbon*CARBON) +
                                  (no_nitro*NITROGEN) + (no_sulfur*SULFUR) +
                                  (no_hydro*HYDROGEN))/total_no;

   /*  Print the average atomic weight.  */
   cout << "The average weight of the atoms in this particular amino "
        << "acid is " << average_atomic_weight << endl;

   /*  Exit program.  */
   return 0;
```

```
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_41                                                    */
/*                                                                         */
/*  This program reads in a positive number and then computes              */
/*  the logarithm of that value to the base 2.                             */

#include <iostream>
#include <cmath>

using namespace std;

int main()
{
   /*  Declare variables.  */
   double  x, answer;

   /*  Enter a positive number.  */
   cout << "Enter a positive number:  ";
   cin >> x;

   /*  Compute the logarithm to base 2.  */
   answer = log(x)/log(2.0);

   /*  Print the answer.  */
   cout << "The logarithm of " << x << " to the base 2 is " << answer
        << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/*  Problem chapter2_42                                                    */
/*                                                                         */
/*  This program reads in a positive number and then computes              */
/*  the logarithm of that value to the base 8.                             */

#include <iostream>
#include <cmath>

using namespace std;

int main()
{
   /*  Declare variables.  */
   double  x, answer;

   /*  Enter a positive number.  */
   cout << "Enter a positive number:  ";
   cin >> x;

   /*  Compute the logarithm to base 8.  */
   answer = log(x)/log(8.0);
```

```
   /*  Print the answer.  */
   cout << "The logarithm of " << x << " to the base 8 is " << answer
        << endl;

   /*  Exit program.  */
   return 0;
}
/*-------------------------------------------------------------------------*/
```



Chapter 2

Simple C++ Programs

Outline

Objectives

1. C++ Program Structure
2. Constant and Variables
3. C++ Classes
4. Building C++ Solutions with IDEs:Xcode
5. C++ Operators
6. Standard Input and Output
7. Building C++ Solutions with IDEs:NetBeans
8. Basic Functions in C++ Standard Library
9. Problem Solving Applied
10. System Limitations

Objectives

Develop problem-solving solutions in C++ containing:

- Simple arithmetic computations
- Information printed on the screen
- User-supplied Information from keyboard
- Programmer-defined data types

C++ Program Structure

```
/*-------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
#include <iostream>  // Required for cout, endl.
#include <cmath>     // Required for sqrt()
using namespace std;
int main() {
// Declare and initialize objects.
   double x1{1}, y1{5}, x2{4}, y2{7},
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.


   cout << "The distance between the two points is "
        << distance << endl;
// Exit program.
return 0;
}
```

C++11 recommended notation for initializing objects.

```
/*------------------------------------------------------------
 * Program chapter1_1
 * This program computes the distance between two points.
 */
                        // Required for cout, endl.
                        // Required for sqrt()


// Declare and initialize objects.


// Compute sides of a right triangle.


// Print distance.


// Exit program.


//------------------------------------------------------------
```

Comments:
- Document the program's purpose
- Help the human reader understand the program
- Are ignored by the compiler
- // comments to end-of line
- /* starts a comment block ending with */

```
#include <iostream>
#include <cmath>

   double x1{1}, y1{5}, x2{4}, y2{7},
   side1, side2, distance;
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is
        << distance << endl;
// Exit program.
return 0;
}
//--------------- ---------- ----------------
```

Preprocessor Directives:
•Give instructions to the preprocessor before the program is compiled.
•Begin with #
#include directives ‘add’ or ‘insert’ the contents of the named file into the program


```
using namespace std;

// Declare and initialize objects.

// Compute sides of a right triangle.

//-----------------------------------------------
```

'using' Directives:
- Tell the compiler to use the library names declared in the specified namespace.
- The 'std', or standard namespace contains C++ language-defined components.


```
int main()
// Declare and initialize objects.
```

`main` function header:
- Defines the starting point (i.e. entry point) for a C++ program
- The keyword '`int`' indicates that the function will return an integer value to the operating system.
- Every C++ program has exactly one function named `main`.

```
{
// Declare and initialize objects.




}
//
```

Code blocks:

- are zero or more C++ declarations and/or statements enclosed within curly braces { }


```
// Declare and initialize objects.
   double x1{1}, y1{5}, x2{4}, y2{7},
   side1, side2, distance;
```

Declarations:

- Define identifiers and allocate memory.
- May also provide initial values for variables.
- Identifiers must be declared before using in a statement.
- C++11 recommends using { } notation.

```
//-----------------------------------------------------------
```

```
// Compute sides of a right triangle.
   side1 = x2 - x1;
   side2 = y2 - y1;
   distance = sqrt(side1*side1 + side2*side2);
// Print distance.
   cout << "The distance between the two points is
        << distance << endl;
// Exit program.
return 0;
```

Statements :

- specify the operations to be performed.

Constants and Variables

- Constants and variables both represent memory locations that we reference in our program solutions.

Constants and Variables

- Constants are objects that store specific data values that can **not** be modified.
  - `10` is an integer constant
  - `4.5` is a floating point constant
  - `"The distance between the two points "` is a string constant
  - 'a' is a character constant
- Variables are named memory locations that store values that **can be modified**.
  - `double x1{1.0}, x2{4.5}, side1;`
  - `side1 = x2 - x1;`
  - `x1, x2` and `side1` are examples of variables that can be modified.

Initial Values

- C++ does not provide initial values for variables.
  - Thus using the value of a variable before it is initialized may result in ‘garbage’.

Memory Snapshots

- Memory ‘snapshots’ are diagrams that show the types and contents of variables at a particular point in time.

```
double x1{1}, y1{5}, x2{4}, y2{7},
       side1, side2, distance;
```

double x1	1.0	double y1	5.0		
double x2	4.0	double y2	7.0		
double side1	?	double side2	?	double distance	?

Valid C++ Identifiers

- Must begin with an alphabetic character or the underscore character '_'
- Alphabetic characters may be either upper or lower case.
  - C++ is CASE SENSITIVE, so 'a' != 'A', etc…
- May contain digits, but not as the first character.
- May be of any length, but the first 31 characters must be unique.
- May NOT be C++ keywords.

C++ Keywords

TABLE 2.1 Keywords

alignas	alignof	and	and_eq	asm
auto	bitand	bitor	bool	break
case	catch	char	char16_t	char32_t
class	compl	const	constexpr	const_cast
continue	decltype	default	delete	do
double	dynamic_cast	else	enum	explicit
extern	false	float	for	friend
goto	if	inline	int	long
mutable	namespace	new	noexcept	not
not_eq	nullptr	operator	or	or_eq
private	protected	public	register	reinterpret_cast
return	short	signed	sizeof	static
static_assert	static_cast	struct	switch	template
this	thread_local	throw	true	try
typedef	typeid	typename	union	unsigned
using	virtual	void	volatile	wchar_t
while	xor	xor_eq		

C++ Identifiers

- Should be carefully chosen to reflect the contents of the object.
  - The name should also reflect the units of measurements when applicable.
- Identifiers must be declared before they may be used.
  - C++ is a ***strongly typed*** programming language.

Common C++ Data Types

- Keyword — Example of a constant
  - `bool` — `true`
  - `char` — `'5'`
  - `int` — `25`
  - `double` — `25.0`
  - `string` — `"hello" //#include<string>`

Declarations

- A type declaration statement defines new identifiers and allocates memory.
- An initial value may be assigned to a memory location at the time an identifier is defined.

Syntax

[modifier] type specifier identifier [{initial value}];
[modifier] type specifier identifier [= initial value];
[modifier] type specifier identifier[(initial value)];

Examples

```
double x1, y1{0}; //C++11
int counter=0;
const int MIN_SIZE=0;
bool error(false);
char comma(',');
```

Symbolic Constants

- A symbolic constant is defined in a declaration statement using the modifier `const`.
- A symbolic constant allocates memory for an object that can **not** be modified during execution of the program. **Any attempt to modify a constant will be flagged as a syntax error by the compiler**.
- A symbolic constant must be initialized in the declaration statement.

Auto Type Specifier

- The C++11 keyword `auto` supports the declaration of an object without specifying a data type, as long as an initializer is provided.
- The data type of the initializer defines the data type of the identifier.

Examples

- `auto x1 = 0; //x1 is type integer`
- `auto comma = ','; //comma is type char`
- `auto y1 = 0.5; //y1 is type double`
- `auto time = x1;  //time is type integer`

`auto` can be useful as data types become more complex, and harder to determine.

Order of Types

- Because different data types have different representations, it may be necessary to convert between data types.
  - Conversion from a lower type to a higher type results in no loss of information.(Example: `double x{1};`)
  - Conversion from higher type to lower type may loose information. (Example:

    `int x(7.7);`) implicit conversion from 'double' to 'int' changes value from 7.7 to 7

High:	**long double**
	double
	float
	long integer
	integer
Low:	**short integer**

C++ Classes

C++ Supports the use of classes to define new data types.

- Definition of a new class type requires a
  - Class Declaration
  - Class Implementation

Class Declarations

- Typically written in a file named “className.h”.
- Begins with keyword `class` followed by the name (identifier) of the new class type.
- Body of the class declaration is a block of code containing

– declaration of data members (attributes)
– method (function) prototypes
– keywords `public, protected`, and `private` are used to control access to data members and methods
– A semicolon must terminate the body of the class declaration. };

Class Implementation

- The class is typically written in a file named "className.cpp"
- File should `#include "className.h"`
- Provides the code to implement class methods.

Class Syntax

Syntax: Class Declaration
//filename:className.h

```
class className
{
    access modifier:
      declaration of attributes
    acccess modifier:
      declaration of methods
};
```

Syntax: Class Implementation
//filename: className.cpp

```
#include "className.h"
definitions of class methods
```

Example: class Declaration
//filename: Point.h

```
class Point
{
  private:
    double xCoord;
    double yCoord;

  public:
  Point(double x, double y);
};
```

Example: class Implementation
//filename: Point.cpp

```
#include "Point.h"

Point::Point (double x, double y)
{
  xCoord = x;
  YCoord = Y;
}
```

Class Methods

- Define the operations that can be performed on class objects.
- A ***constructor*** is a special method that is executed when objects of the class type are declared (***instantiated***).
  - Constructors have the same name as the class.
  - A class may define multiple constructors to allow greater flexibility in creating objects.
    - The default constructor has no parameters.
    - Parameterized constructors provide initial values for data members.

Using a Class


```
Usage:
#include "Point.h"
...
int main()
{
  Point p1(1.5, 2.7);
...
```

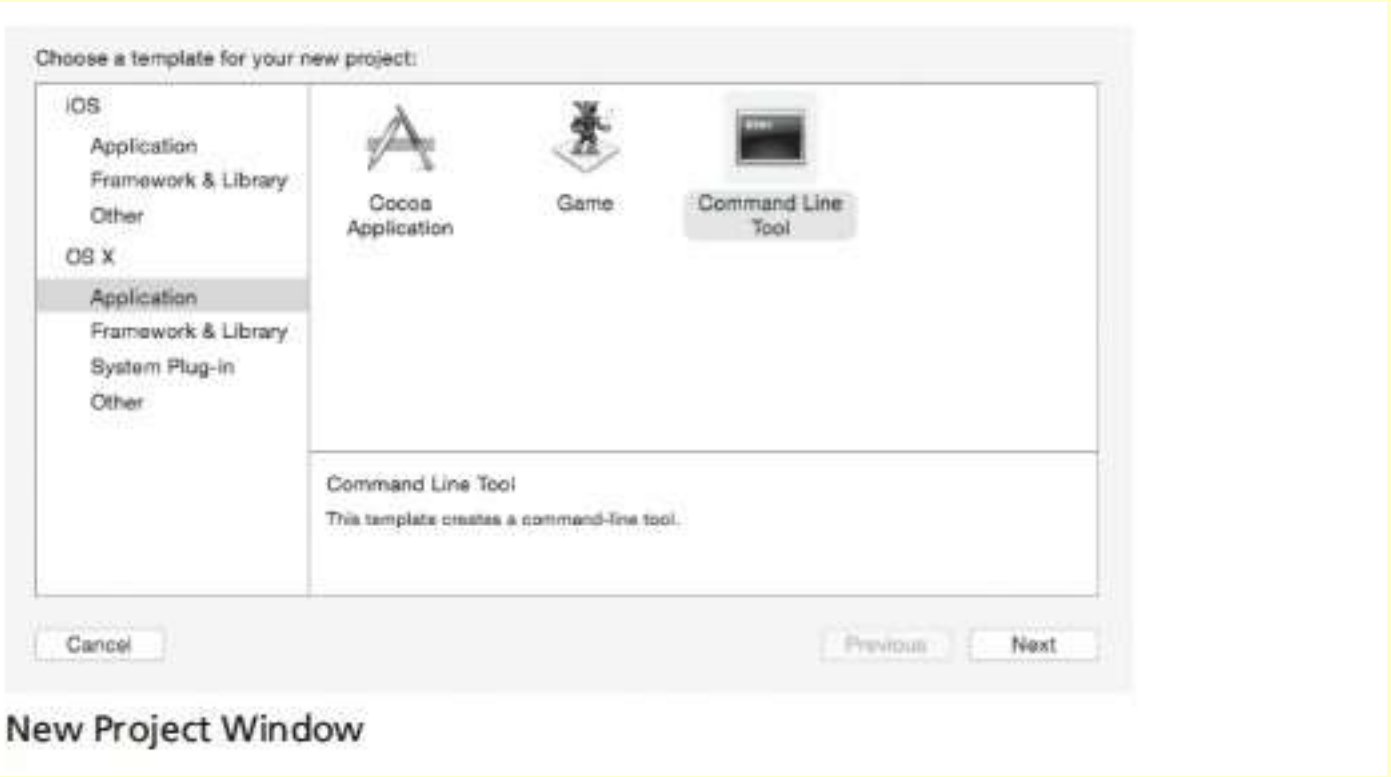
- Once a class is defined, you may use the class name as a type specifier.
  - You must include the class declaration (i.e. header file)
  - You must link to the class implementation (i.e. `.cpp` file)

Integrated Development Environments (IDEs)

- IDEs are software packages designed to facility the development of software solutions.
- IDEs include:
  - Code editors
  - Compiler
  - Debugger
  - Testing tools
  - Many additional helpful tools…

Xcode

The Xcode IDE was developed by Apple for the development of applications that run on Macs, iPads, and iPhones. Xcode is available as a free download at https://developer.apple.com/ .



New Project Window

C++ Operators

- Assignment Operator
- Arithmetic Operators
- Increment and Decrement Operators
- Abbreviated Arithmetic Operators

Assignment Operator

- The assignment operator (=) is used in C++ to assign a value to a memory location.
- The assignment statement:

```
x1 = 1.0;
```

  – assigns the value `1.0` to the variable `x1`.
  – Thus, the value `1.0` is stored in the memory location associated with the identifier `x1`. (Note: `x1` must have been previously declared.)

Assignment Statements

Syntax

```
identifier = expression;
```

Examples

```
x1 = y1;
counter = 0;
counter = counter + 1;
```

Assignment operator(=) should not be confused with equality operator(==).

Arithmetic Expressions

- Expressions used in assignment statements for numeric variables may be literal constants (e.g. `x1 = 10.4;`), other variables (e.g. `x2= x1;`), or compound expressions involving arithmetic operators (e.g. `x1 = -3.4*x2 + 10.4`).

Arithmetic Operators

- Addition **+**
- Subtraction **-**
- Multiplication *****
- Division ***/***
- Modulus **%**
  - Modulus returns remainder of division between two *integers*
  - Example

    5%2 returns a value of 1

Operator Basics

- The five operators (* / % + -) are ***binary operators*** - operators that require two arguments (i.e. operands).
- C++ also includes ***unary operators -*** operators that require only a single argument.
  - For example, the minus sign preceding an expression, as in (y = `-x2`), is a unary operator.

Integer Division

- Division between two integers results in an integer.
- The result is truncated, not rounded
- Example:

  The expression **5/3** evaluates to 1

  The expression **3/6** evaluates to 0

Mixed Operations

- Binary operations on two values of same type yield a value of that type (e.g. dividing two integers results in an integer).
- Binary operations between values of different types is a ***mixed operation***.
  - Value of the lower type must be converted to the higher type before performing operation.
  - Result is of the higher type.

Casting

- The cast operator.
  - The cast operator is a unary operator that requests that the **value** of the operand be cast, or changed, to a new type for the next computation. **The type of the operand is not affected.**
- Example:

```
int count{10}, sum{55};
double average;
average = (double)sum/count;
```

Memory snapshot:

int count	10
int sum	55
double average	5.5

Overflow and Underflow

- Overflow
  - answer too large to store

  Example: using 16 bits for integers

  ```
  result = 32000+532;
  ```
- Exponent overflow
  - answer's exponent is too large

  Example: using float, with exponent range –38 to 38

  ```
  result = 3.25e28 * 1.0e15;
  ```
- Exponent underflow
  - answer's exponent too small

  Example: using float, with exponent range –38 to 38

  ```
  result = 3.25e-28 *1.0e-15;
  ```

Increment and Decrement Operators

- Unary Operators
- Increment Operator **++**
  - post increment **x++;**
  - pre increment **++x;**
- Decrement Operator **--**
  - post decrement **x--;**
  - pre decrement **--x;**
- For example, assume k=5 prior to executing each of the following statements.
  - m = ++k; // m and k are 6 after execution
  - n = k- -; // n is 5 and k is 4 after execution

Abbreviated Assignment Operators

operator	example	equivalent statement
+=	**x+=2;**	x=x+2;
-=	**x-=2;**	x=x-2;
*=	**x*=y;**	x=x*y;
/=	**x/=y;**	x=x/y;
%=	**x%=y;**	x=x%y;

Operator Precedence

Precedence	Operator	Associativity
1	Parenthesis: ()	Innermost First
2	Unary operators: + - ++ -- (**type**)	Right to left
3	Binary operators: * / %	Left to right
4	Binary operators: + -	Left to right
5	Assignment Operators = += -= *= /= %=	Right to left

Standard Input/Output

`cin` Standard input, pronounced “c in”

`cout` Standard output, pronounced “c out”

Standard Output - cout

- `cout` is an `ostream` object, defined in the header file `iostream`
- `cout` is defined to stream data to standard output (the display)
- We use the output operator `<<` with `cout` to output the value of an expression.

  General Form:
  `cout` **<<** expression **<<** expression**;**

Note: An expression is a C++ constant, identifier, formula, or function call.

Standard Input - cin

- `cin` is an `istream` object defined in the header file `iostream`
- `cin` is defined to stream data from standard input (the keyboard)
- We use the input operator `>>` with `cin` to assign values to variables
  - General Form
    `cin >>` *identifier* `>>` *identifier*;
- **Note: Data entered from the keyboard must be compatible with the data type of the variable.**

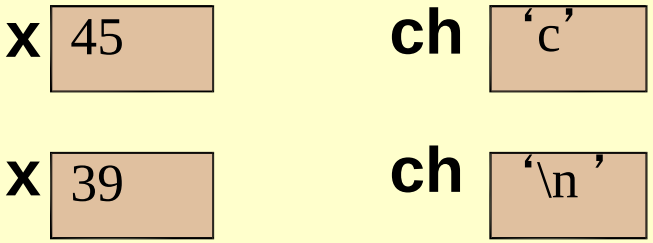
Characters and Input

- The input operator >> skips all whitespace characters.
- The get() method gets the next character.
- Example:

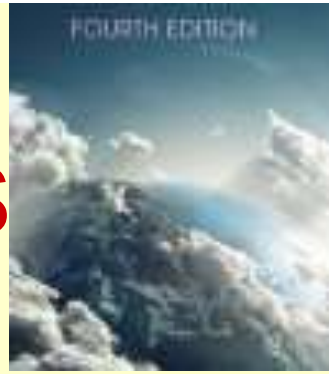
```
int x;
char ch;
cin >> x >> ch;
cin >> x;
cin.get(ch);
```

Input stream:
45 c
39
b

Memory Snapshot



Manipulators and Methods

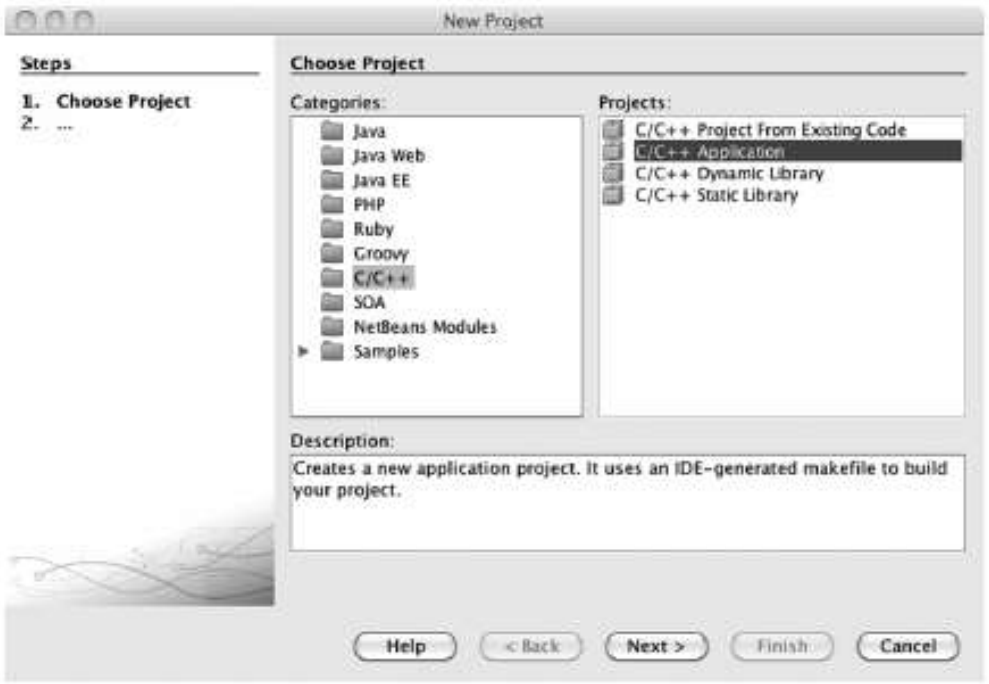


- `endl` – places a newline character in the output buffer ***and flushes the buffer.***
- setf() and unsetf()

Flag	Meaning
ios::showpoint	display the decimal point
ios::fixed	fixed decimal notation
ios::scientific	scientific notation
Ios::setprecision(n)	set the number of significant digits to be printed to the integer value n
Ios::setw(n)	set the minimum number of columns for printing the next value to the integer value n
ios::right	right justification
ios::left	left justification

Building C++ Solutions with IDEs:NetBeans

The NetBeans IDE is open source software that provides a development environment for multiple languages including Java, C and C++.



Basic Functions in C++ Standard Library

Math Functions #include<cmath>

fabs(x)	computes absolute value of x
sqrt(x)	computes square root of x, where x >=0
pow(x,y)	computes x^y
ceil(x)	nearest integer larger than x
floor(x)	nearest integer smaller than x
exp(x)	computes e^x
log(x)	computes ln x, where x >0
log10(x)	computes $\log_{10}x$, where x>0

Trigonometric Functions

sin(x)	sine of x, where x is in radians
cos(x)	cosine of x, where x is in radians
tan(x)	tangent of x, where x is in radians
asin(x)	This function computes the arcsine, or inverse sine, of x, where x must be in the range [−1, 1]. The function returns an angle in radians in the range [−π/2, π/2].
acos(x)	This function computes the arccosine, or inverse cosine, of x, where x must be in the range [−1, 1]. The function returns an angle in radians in the range [0, π].
atan(x)	This function computes the arctangent, or inverse tangent, of x. The function returns an angle in radians in the range [−π/2, π/2].
atan2(y,x)	This function computes the arctangent or inverse tangent of the value y/x. The function returns an angle in radians in the range [−π, π].

Common Functions Defined in <cctype>

isalpha(ch)	Returns true if ch is an upper or lower case letter.
isdigit(ch)	Returns true if ch is a decimal digit
isspace(ch)	Returns true if ch is a whitespace character.
islower(ch)	Returns true if ch is an lower case letter.
isupper(ch)	Returns true if ch is an upper case letter.
tolower(ch)	Returns the lowercase version of ch if ch is an uppercase character, returns ch otherwise.
toupper(ch)	Returns the uppercase version of ch if ch is a lowercase character, returns ch otherwise.

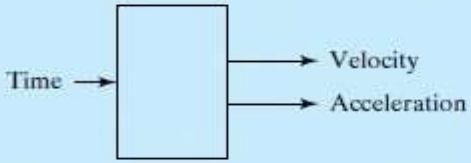
Problem Solving Applied

1. PROBLEM STATEMENT

Compute the new velocity and acceleration of the aircraft after a change in power level.

2. INPUT/OUTPUT DESCRIPTION

The following diagram shows that the input to the program is a time value, and that the output of the program is the pair of new velocity and acceleration values. The built-in data type `double` can be used to represent these values.



3. HAND EXAMPLE

Suppose that the new time value is 50 seconds. Using the equations given for the velocity and accelerations, we can compute these values:

Velocity = 208.3 m/s;
Acceleration = 0.31 m/s^2.

4. ALGORITHM DEVELOPMENT

The first step in the development of an algorithm is the decomposition of the problem solution into a set of sequentially executed steps:

Decomposition Outline

1. Read new time value.
2. Compute corresponding velocity and acceleration values.
3. Print new velocity and acceleration.

Because this program is a very simple program, we can convert the decomposition directly to C++.


```
/*-------------------------------------------------------------*/
/*  Program chapter2_6                                         */
/*                                                             */
/*  This program estimates new velocity and                    */
/*  acceleration values for a specified time.                  */

#include<iostream>  //Required for cin,cout
#include<iomanip>   //Required for setprecision(), setw()
#include<cmath>     //Required for pow()
using namespace std;

int main()
{
   //  Declare objects.
   double time, velocity, acceleration;

   //  Get time value from the keyboard.
   cout << "Enter new time value in seconds: \n";
   cin >> time;

   //  Compute velocity and acceleration.
   velocity = 0.00001*pow(time,3) - 0.00488*pow(time,2)
              + 0.75795*time + 181.3566;
   acceleration = 3 - 0.000062*velocity*velocity;
   //  Print velocity and acceleration.
   cout << fixed << setprecision(3);
   cout << "Velocity = " << setw(10)
        << velocity << " m/s" << endl;
   cout << "Acceleration = " << setw( 14)
        << acceleration << "m/s^2" << endl;

   //  Exit program.
   return 0;
}
/*-------------------------------------------------------------*/
```

System Limitations

- C++ standards do not specify limitations of data types – they are compiler-specific.
- C++ does provide standard methods of accessing the limits of the compiler:
  - <climits> defines ranges of integer types.
  - <cfloat> defines ranges of floating-point types.
  - the sizeof(**type**) function returns the memory size of the type, in bytes.