Variable and Constant Definitions

```
 Type    Name         Initial value
int cans_per_pack = 6;

const double CAN_VOLUME = 0.335;
```

Mathematical Operations

#include <cmath>

`pow(x, y)`	Raising to a power x^y
`sqrt(x)`	Square root $\sqrt{x}$
`log10(x)`	Decimal log $\log_{10}(x)$
`abs(x)`	Absolute value $\lvert x \rvert$
`sin(x)` `cos(x)` `tan(x)`	Sine, cosine, tangent of x (x in radians)

Selected Operators and Their Precedence

(See Appendix B for the complete list.)

`[]`	Array element access
`++ -- !`	Increment, decrement, Boolean *not*
`* / %`	Multiplication, division, remainder
`+ -`	Addition, subtraction
`< <= > >=`	Comparisons
`== !=`	Equal, not equal
`&&`	Boolean *and*
`||`	Boolean *or*
`=`	Assignment

Loop Statements

```
                 Condition
while (balance < TARGET)
{
   year++;                                     Executed
   balance = balance * (1 + rate / 100);       while condition
}                                              is true

    Initialization  Condition  Update
for (int i = 0; i < 10; i++)
{
   cout << i << endl;
}

                 Loop body executed
do                  at least once
{
   cout << "Enter a positive integer: ";
   cin >> input;
}
while (input <= 0);
```

Conditional Statement

```
                 Condition
if (floor >= 13)
{                                Executed when
   actual_floor = floor - 1;     condition is true
}
else if (floor >= 0)             Second condition (optional)
{
   actual_floor = floor;
}
else
{                                            Executed when all
   cout << "Floor negative" << endl;         conditions are false
}                                            (optional)
```

String Operations

```
#include <string>
string s = "Hello";
int n = s.length(); // 5
string t = s.substr(1, 3); // "ell"
string c = s.substr(2, 1); // "l"
char ch = s[2]; // 'l'
for (int i = 0; i < s.length(); i++)
{
   string c = s.substr(i, 1);
   or char ch = s[i];
   Process c or ch
}
```

Function Definitions

```
  Return type         Parameter type and name
double cube_volume(double side_length)
{
   double vol = side_length * side_length * side_length;
   return vol;
}                     Exits function and returns result.
Reference parameter
void deposit(double& balance, double amount)
{
   balance = balance + amount;
}
                Modifies supplied argument
```

Arrays

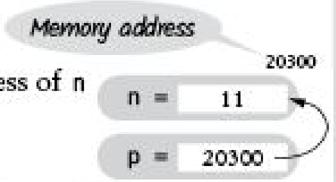
```
 Element type   Length
int numbers[5];
int squares[] = { 0, 1, 4, 9, 16 };
int magic_square[4][4] =
{
   { 16, 3, 2, 13 },
   { 5, 10, 11, 8 },
   { 9, 6, 7, 12 },
   { 4, 15, 14, 1 }
};

for (int i = 0; i < size; i++)
{
   Process numbers[i]
}
```

Vectors

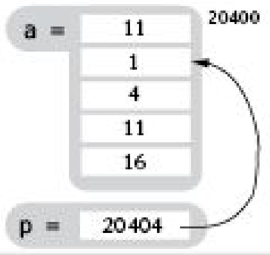
```
#include<vector>   Element type     Initial values (C++ 11)

vector<int> values = { 0, 1, 4, 9, 16 };
                                        Initially empty
vector<string> names;
                                        Add elements to the end
names.push_back("Ann");
names.push_back("Cindy"); // names.size() is now 2

names.pop_back(); // Removes last element

names[0] = "Beth"; // Use [] for element access
```

Pointers

```
int n = 10;
int* p = &n; // p set to address of n
*p = 11; // n is now 11
```



```
int a[5] = { 0, 1, 4, 9, 16 };
p = a; // p points to start of a
*p = 11; // a[0] is now 11
p++; // p points to a[1]
p[2] = 11; // a[3] is now 11
```



Input and Output

```
#include <iostream>
cin >> x; // x can be int, double, string
cout << x;
while (cin >> x) { Process x }
if (cin.fail()) // Previous input failed
#include <fstream>
string filename = ...;
ifstream in(filename);
ofstream out("output.txt");
string line; getline(in, line);
char ch; in.get(ch);
```

Static Variables

```
void increment_print() {
  static int s_value = 0; //static duration
  s_value++;
  cout << s_value << '\n';
} //s_value is not destroyed, but goes out of scope
int main() {
  increment_print(); //1
  increment_print(); //2
}
```

Static Data Members

```
class Item {
private:
   int m_id;
   static int s_id_counter;
public:
   Item() {
      m_id = s_id_counter++;
   }
   int get_id() const {
      return m_id;
   }
};
int Item::s_id_counter = 1;
int main() { //
   Item first;
   Item second;
   cout << first.get_id();   //1
   cout << second.get_id();//2
}
```

Range-based for Loop

```
                 An array, vector, or other container (C++ 11)
for (int v : values)
{
   cout << v << endl;
}
```

Output Manipulators

`#include <iomanip>`

`endl`	Output new line
`fixed`	Fixed format for floating-point
`setprecision(n)`	Number of digits after decimal point for fixed format
`setw(n)`	Field width for the next item
`left`	Left alignment (use for strings)
`right`	Right alignment (default)
`setfill(ch)`	Fill character (default: space)

Enumerations, Switch Statement

```
enum Color { RED, GREEN, BLUE };
Color my_color = RED;

switch (my_color) {
  case RED :
    cout << "red";   break;
  case GREEN:
    cout << "green"; break;
  case BLUE :
    cout << "blue";  break;
}
```

Class Definition

```
class BankAccount
{
public:
   BankAccount(double amount);     Constructor declaration
   void deposit(double amount);    Member function declaration
   double get_balance() const;     Accessor member function
   ...
private:                Data member
   double balance;
};

void BankAccount::deposit(double amount)     Member function
{                                            definition
   balance = balance + amount;
}
```

Inheritance

```
                    Derived class          Base class
class CheckingAccount : public BankAccount
{
public:                                    Member function
   void deposit(double amount);            overrides base class
private:
   int transactions;          Added data member
};                            in derived class

void CheckingAccount::deposit(double amount)
{
   BankAccount::deposit(amount);           Calls base class
   transactions++;                         member function
}
```