1. **How to set the exact decimal points**

```
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.precision(2);
```

2. **Random number between (0,1)**

(1)(RAND_MAX–rand())/static_cast<double>(RAND_MAX)

(2) count = (rand() % 11) + 2 // [2,12]

(3) swith, random on other types

```
string randomHobbit( ) {
     int r = 1+rand()%4;
     switch (r) {
          case 1: return "Frodo"; break;
          case 2: return "Sam"; break;
          case 3: return "Pippin"; break;
          case 4: return "Merry"; break;
     }
}
```

```
// a-z, [97,122], get random 4 letter word
string getWord() {
    string s = "abcd";
    for (int i = 0; i < 4; i++)
        s[i] = (char) (97+rand()%(122-97+1));
    return s;
}
```

4. **Nearest to an integer (ceil, floor)**

```
double d = … some value …;
int i = static_cast<int>(floor(d+0.5));
```

5. **#include<>**

iostream like: cin, cout

cmath like: sqrt, exp(e^x), floor, log(ln)

cstdlib like: exit, rand(), srand(unsigned int)

cctype:

bool isdigit(char)	Digit
bool isupper(char)	Upper case
bool isspace(char)	White space
bool ispunct(char)	Punctuation
salpha(char);	Alphabet letter
isalnum(char);	Alpha or digit
isspace(char);	Any whitespace
ispunct(char);	Punctuation
int toupper(char)	To upper case

string String capabilities

6. **Variable Name**

(1) Cannot use keywords/reserved word

(2) Must start with either a letter or _,

(3) Must all be letters, digits, or_

7. **Precedence Levels**

- Postfix increment and decrement
- Prefix increment and decrement, Unary + and -, Not (!)
- (*, /, %)
- (+, -)
- (<<, >>)
- (<, >, <=, >=)
- (==, !=)
- (&&)
- (| |)
- (=)

8. **Dangling Else**

(1) Else goes with nearest if;

(2) 如果有好几个 if ， 要记得看是不是都能进

9. **Switch (cannot switch string type)**

```
switch(integer expression)
{
    case 1:    statement1;
               break;
     default:  statement3;
}
```

10. **Do . . . While Statement**

Semicolon required on *while* line

```
do {
// Code block
} while (condition);
```

11. **While/Do...While Similarity**

Both require initialization before loop

Both require explicit update within loop

12. **Class String**

1) cin >> string (Reads one word: Up to white space. Stops on white space character)
2) **getline(cin,string);**(Reads one complete line. Stops on first character of next line)
3) **Console Input: cin.ignore()**

Skip up to 10000 characters to newline

cin.ignore(10000, '\n');

4) **String Length:** (return int)

name.length() or name.size()

5) **String Access:**

name[3] or name.at(3)

6) **String Comparison:** space<0<A<a
7) **StringConcatenation:** name1+name2
8) **Replacement:**

name1.replace(n1,n2,name2,n3,n4);

[n1and n3 are starting pos of name1and name2, n2 and n4 are the length of name1 and name2]

9) **Find:** name.find(sth, n1);

[sth is string or char. n1 is the starting pos]

10) Substring: name.substr(m, n)

[m is starting pos. n is length]

5.) *[8 points]* Write a function **`isSubstring`** that takes two strings as input and returns true if the second string is contained in the first string. It should return false if the second string is not contained in the first string. Note that if the second string is longer than the first string, you can immediately return false. An example is shown below.

```
isSubstring("Gandalf", "dal")       returns true
isSubstring("Gandalf", "doh")       returns false
isSubstring("Gandalf", "Gandalf Rocks")   returns false
```

```
bool isSubstring (string s1, string s2) {
     if (s2.length() > s1.length())
          return false;
     for (int i=0; i <= s1.length()-s2.length(); i++)
          if (s1.substr(i,s2.length()) == s2)
               return true;
     return false;
}
```

```
string rotate(string s, int k)
{
    if (k < 0 || s.empty())
       return s;

    int toRotate = k % s.size();
    return s.substr(s.size() - toRotate) +
           s.substr(0, s.size() - toRotate);
}
```

```
bool isHebrew(string word)
{
   for (int i = 0; i < word.size(); i++)
   {
      if (word[i] == 'a' || word[i] == 'e' || word[i] == 'i' ||
          word[i] == 'o' || word[i] == 'u')
         return false;
   }
   return true;
}
```

b)

```
int hebrew(string words[], int n)
{
   if (n < 0)
      return -1;

   int count = 0;
   for (int i = 0; i < n; i++)
   {
      if (isHebrew(words[i]))
      {
         words[count] = words[i];
         count++;
      }
   }
   return count;
}
```

11) String Insertion:

string st1 = "One string";

string st2 = " more";

string st3 = st1.insert(3, st2, 1, 2)

("Onemo string")

12) Erase:

name.erase(n1,n2); // [n1 is starting pos. n2 is length]

```
string Q = "ABCDEFG";
cout << Q + Q.substr(2,4) << "\n";
Q.erase(3,2);
cout << Q << "\n";
Q.insert(2,"123");
cout << Q;

ABCDEFGCDEF
ABCFG
AB123CFG
```

17. C string

- ***NULL*** character is terminator ('\0')
- Convert to C++ string, just use =
- Convert to C string, char* C++str.c_str() (return C string has ***NULL***)
- ***strlen***(string) (not include null)
- ***strcpy***(to_string, from_string, limit)
- ***strcat***(to_string, from_string, limit)
- int ***strcmp***(string1, string2, limit) (return value <0, st1< st2, others same)
- C String Output just use cout << st;

13. **Array Parameters （call by reference）**

(1) Only array name is used an argument

(2) array name is a pointer to its first element and it is a **const pointer**.

(3) **a[n]** is really equivalent to ***(a+n)**

(4) 2-D Array: int a[row][collum];

int grid [][3]; //second dimension must be specified

(5) array size must be a constant.

Eg. const int SIZE = 20; int arr[SIZE] = {0};

(6) array size not known until runtime, dynamic array

Eg. int size = rand()%10; int* Car = new Car[size];

18. Pointers

(1) always initialize before use

Eg. string c = "Hello"; string* p = **&**c;

char* suit[5] = {"hello", "lala"};

(2) const int* p; // pointer to a const

(3) int* const p; // a const pointer

(4) **typedef** int* ptr;

(5) ar[1][2] 或者 *(ar[1] + 2) 或者 *(*(ar+1)+2)

19. **Dynamic Data**

(1) int* p1 = new int; int* p2 = new int(14);

delete p1; p1 = NULL;

(2) int* p = new int[size]; **delete [ ]** p;

(3) Car* arr[MAXSIZE]; // array of pointers to Car

```
int* p1 = new int[10];
int* p2[15];
for (int i = 0; i < 15; i++)
   p2[i] = new int[5];
int** p3 = new int*[5];
for (int i = 0; i < 5; i++)
   p3[i] = new int;
int* p4 = new int;
int* temp = p4;
p4 = p1;
p1 = temp;
```

```
delete p1;    // "delete temp;" works too.

for (int i = 0; i < 5; i++)
   delete p3[i];
delete[] p3;  // This must happen AFTER
              // the above for loop.

for (int i = 0; i < 15; i++)
   delete[] p2[i];

delete[] p4;
```

20. Structs, Classes(end with;)

(1) Const parameter, prevents the parameter assigned a value

(2) Const after a function, it prevents changing any of the member data

(3) Static variables must be initialized once **outside** the class definition

(4) Static functions cannot use any nonstatic data

members or any nonstatic member functions (**Static** used in function definition, not function declaration)

(5) The ***this*** pointer is a constant pointer: it cannot be changed (it cannot be used in static functions)

21Assert

include <cassert>

assert(expression); (true, nothing happens)

22Default Arguments in Calls

1) Can only be omitted at end (from right)

2) Default values will be used for arguments which are omitted from a call

/*11 line N shape*/

```
#include <iostream>
using namespace std;
int main () {
    for (int i = 0; i < 11; ++i){
        cout << "N";
        for (int j = 1; j < 10; ++j){
            if (i == j) cout << "N";
            else cout << " ";
        }
        cout << "N" << endl;
    }
}
```

/*11 line Cross Shape*/

```
for (int i = 0; i < 11; ++i){
    if (i == 5) {
        for (int j = 0; j < 11; ++j){
            cout << "+";
            }
            cout << endl;
            }
            else {
                for (int j = 0; j < 11; ++j){
                    if (j == 5) cout << "+";
                    else cout << " ";
                }
                cout << endl;
            }
        }
```

/*Triangle*/

```
#include <iostream>
using namespace std;
int main() {
    int num = 0;
    cout << "enter a number";
    cin >> num;
    for (int i = 1; i <= num; i++){
        int c = num + i - 1;
        for (int j = 0; j < c; j++)
            if (j < num - i)
                cout << " ";
            else cout << "*";
        cout << endl;
    }
}
```

// 最大公约数

```
int BiggestCommonFactor(int a,int b) {
    //int result = 0;
    int min = (a < b)? a : b;
    for (int i = min-1; i >= 2; i--) {
        if ((a % i == 0) && (b % i == 0))
            return i;
    }
    return -1;
}
```

11.) *[7 points]* A prime number is a number that is only divisible by itself and 1. Write a boolean function `isPrime` that takes an integer as input and returns true if the number is prime. You may assume the input is a positive number larger than one.

```
bool isPrime (int N) {
     for (int i=2; i<N; i++)
          if (N%i == 0)
               return false;
     return true;
}
```

// search a value in 2D array

```
// Return index of x if found, or -1 if not
int search ( int A[], int N, int x ){
    for ( int m = 0; m < N; m++ )
        if ( A[m] == x )
            return m;
    return -1;
}
```

/* delete duplicates in sorted array*/

```
int deleteDuplicates(string a[], int size) {
    // if the parameter size is not reasonable, return -1
    if (size < 0)
        return -1;
    //
    int i = 1, j = 0;
    // detect the whole array staring from index i = 1
    for (i = 1; i < size; i++){
        // if a[i] is not equal to a[j]
        if (a[i] != a[j]) {
            a[j+1] = a[i]; // assign a[i] to a[j+1]
            j++; // increment index j
        }
    }
    // return the number of remianing elements
    return j+1;
}
```

```
/* merge two non-increasing arrays.*/
int merge(const string a1[], int size1, const string a2[], int size2, string result[],
    if ((size1 < 0) || (size2 < 0) || ((size1+size2) > size))
        return -1;
    for (int i = 0; i < (size1-1); i++) {
        if (a1[i] < a1[i+1])
            return -1;
    }
    for (int i = 0; i < (size2-1); i++) {
        if (a2[i] < a2[i+1])
            return -1;
    }
    int j = 0, k = 0; // j is the index of string a1, k is the index of string a2, both
    for (int i = 0; i < size1+size2; i++) {
        // if both a1 and a2 elements have not been used up
        if ((j < size1) && (k < size2)) {
            if (a1[j] > a2[k]) {
                result[i] = a1[j];
                j++;
            }
            else {
                result[i] = a2[k];
                k++;
            }
        }
        // if a1 first used up its elements, copy the left elements in a2 to the result
        else if (j == size1) {
            result[i] = a2[k];
            k++;
        }
        else {
            result[i] = a1[j];
            j++;
        }
    }
    return size1+size2;
}
```

```
/* Eliminate the element at loc
 and move it to the starting position of the array, swap*/
int moveToStart(string a[], int n, int loc) {
    // if the parameter is not reasonable, return -1
    if ((n < 0) || (loc >= n))
        return -1;
    string temp;
    for (int i = 0; i < loc; i++) {
        temp = a[loc];
        a[loc] = a[i];
        a[i] = temp;
    }
    // return the location
    return loc;
}
```

```
// function 5 sort list, using bubble sort
bool PHB_list_sort(int *list, const int *len) {
    if ((list == NULL) || (*len <= 0)) {
        return false;
    }
    int compare_time = *len -1;
    //set a pointer current, first points to the head of the array
    int *current = list;
    while (compare_time > 0){
        for (int i = 0; i < compare_time; i++) {
            if (*current > *(current+1)){
                    int temp = *current;
                    *current = *(current+1);
                    *(current+1) = temp;
                }
            current++;
        }
        current = list;// reset the pointer current to the head of arra
        compare_time--;
    }
    return true;
}
```

/* stock market*/

```
void printSummary(Stock stocks[], int n)
{
    cout << "STOCK" << "\t" << "TOTAL VALUE" << "\n";
    cout << "=====" << "\t" << "============" << "\n";

    for (int i = 0; i < n; i++)
        printEntry(stocks[i]);

    cout << "=====" << "\t" << "============" << "\n";
    cout << "TOTAL" << "\t" << sumValues(stocks, n) << endl;
}

void printEntry(Stock stock) {
    cout << stock.symbol << "\t" << stock.numUnits * stock.value
}

double sumValues(Stock stocks[], int n)
{
    double sum = 0;
    for (int i = 0; i < n; i++)
        sum += stocks[i].numUnits * stocks[i].value;
    return sum;
}
```

/* 把零钱换算成整的，输出*/

```
CoinMoon::CoinMoon()
: totalInCents(0)
{}

void CoinMoon::addDollar()
{
  totalInCents += 100;
}

void CoinMoon::addQuarter()
{
  totalInCents += 25;
}

void CoinMoon::addDime()
{
  totalInCents += 10;
}

void CoinMoon::addNickel()
{
  totalInCents += 5;
}
```

```
void CoinMoon::giveMeBills()
{
  int dollars = totalInCents / 1
  totalInCents %= 100;
  int quarters = totalInCents /
  totalInCents %= 25;
  int dimes = totalInCents / 10;
  totalInCents %= 10;
  int nickels = totalInCents / 5
  totalInCents %= 5;
  int pennies = totalInCents;

  cout << dollars << " dollar bi
       << endl;
  cout << quarters << " quarter(
       << endl;
  cout << dimes << " dime(s)" <<
  cout << nickels << " nickel(s)
       << endl;
  cout << pennies << " pennie(s)
       << endl;

  totalInCents = 0;
}
```

/*Hotel room number*/

```
bool Hotel::validNum(int roomNum)
{
    int floor = roomNum / 100;
    int room = roomNum % 100;

    return (floor >= 0 && floor < FLOORS) && (room >= 0 && room < ROOMSPERFLOOR);
}

bool Hotel::changeState(int roomNum, char from, char to)
{
    // Change the room specified by roomNum from one state to another.
    if (validNum(roomNum) && m_rooms[roomNum / 100][roomNum % 100] == from)
    {
        m_rooms[roomNum / 100][roomNum % 100] = to;
        return true;
    }
    return false;
}
```

/*HW6 Watch*/

```
class CS31Watch{
    public:
        void setTime (int, int, int);
        void getTime (int &, int &, int &) const;
        void printTime () const;
        void incrementSeconds ();
        void incrementMinutes ();
        void incrementHours ();
        bool equalTime(const CS31Watch &) const;
        CS31Watch(int, int, int);
        CS31Watch();
        ~CS31Watch();
    private:
        int hr;     // number of hours
        int min;    // number of minutes
        int sec;    // number of seconds
};
```

```
// incrementHours()
//
void CS31Watch::incrementHours() {
    hr++;
    if (hr > 23)
        hr = 0;
}

//
// incrementMinutes()
//
```

```
bool CS31Watch::equalTime(const CS31Watch * anotherClock ) const {
   return ( (hr == anotherClock->hr)
          && (min == anotherClock->min)
          && (sec == anotherClock->sec));
}
```

```
void CS31Watch::printTime() const {
    // print out the hour digits
    if (hr < 10)
        cout << "0";
    cout << hr << ":";

    // print out the minute digits
    if (min < 10)
        cout << "0";
    cout << min << ":";

    // print out the second digits
    if (sec < 10)
        cout << "0";

    return;
}
```

```
void CS31Watch::setTime(int hours, int minutes, int seconds) {
// calculate the hour digits
    if ( 0 <= hours && hours < 24)
        hr = hours;
    else
        hr = 0;

    // calculate the minute digits
    if ( 0 <= minutes && minutes < 60)
        min = minutes;
    else
        min = 0;

    // calculate the second digits
    if ( 0 <= seconds && seconds < 60)
        sec = seconds;
    else
        sec = 0;

    return;
}
```