| | | |
|---|---|---|
| Basic C++ Operations | Variables | int, double, char, bool (-1 is true)<br>int height, width = 5;   //height is unintialized<br><br>Identifier: Start with A-Z, a-z, _; Rest: A-Z, a-z, _, **0-9**<br>Case sensitive; Cannot be a keyword (cin, cout allowed)<br>Declaration: int a,b,c;     Initialization: int a=2, c=1.5e7;<br>a=b vs a==b: a=b returns value of a<br>const: Never change, otherwise compilation error<br>Modulus: %: Only apply on int: Take abs of ending, Save sign of front<br>Division: /: Truncates any frational part: 8/3=2; Solution: .0, LL<br>Type conversion: Narrow -> Wide or Wide -> Narrow (with warning)<br><br>Order of Operations: ! ++ --, * / %, + -, < <= > >=, == !=, &&, \|\|<br>Prefix/Postfix: Increase before variable is used, after otherwise<br>^ Works on double<br>Shortcut operator<br><br>Variables declared in global scope are initialized to 0<br>RAND_MAX: 32767<br>rand()<br>srand(time(NULL)); #include <ctime> |
| | Arrays | int arr[100];<br>int a[7]={0,1,2,3,4};, a[6]=0<br>int a,arr[]={1,2,3}; is valid<br>Pass array as parameter: Auto pass by reference  void f(int arr[]); |
| | | 2D array: int arr[3][2]={{1,2},{3,4},{5,6}};   //row, then col<br>2D array parameter: Size of first dimension is not given, but remaining must be give: void f (int arr[][2]);<br>Array out of range: No errors |
| | Flow of Control | if/else: Else is always associated with the nearest if; mind { }<br>for: for (int i=0; i<10; i=i+1){   }<br>while: while (true) {   } //never put a ; after while()<br>do-while: Do at least once   do {   } while (true); //inputting data<br>break;/continue;<br>switch (label) {  case 1: break; default: }   const,no string/double<br>default need not at the last, is optional |
| | Input Output | Standard IO: #include <iostream><br>cout << “Hello” << x << endl;    Escape: \t, \”, \’. \?, \\, \n<br>input 1.2 3:    cin >> x >> y    // x=1; y=0.2<br>cin.get(x): read space or new line character    or x=cin.get();<br>getline(cin, s): read a whole line (without “\n”, with init. space) |
| | | File IO: #include <fstream><br>ifstream testmarks;<br>testmarks.open(“data.txt”);<br>testmarks >> x;<br>Keep on reading: while (testmarks >> val){ cout << marks; } |
| | | ofstream results;<br>results.open(“res.txt”, ios::app); append, otherwise erase (create)<br>results << x; |
| | | Remember close: testmarks.close();<br>Check if successful or not: if (testmarks.fail()) {return 0;} |

## C++ Classes and Advanced Usage

### Strings

```
#include <string>    string msg1,mgs2,msg3  msg1=”hello”  msg2=msg1;
Concatenation: +   //one of them must be a variable, a=”b”+”c” error
strcmp: Dictionary order: ‘A’<’a’,  ‘fast’ < ‘fastest’
return –ve int if s<t, +ve if s>t, 0 if s==t
```

```
s.length(): return simple length, no \0
s.empty(): return bool (true if empty)  // mind “” vs “ “
s.substr(pos, n): return string start from pos (inc.), total length<=n
s.substr(pos): return string start from pos (inc.) to end of str
s.find(t): return pos of first occurrence of t in s,
s.find(t,p): return pos of first occurrence of t in s where pos>=p
             return string::npos if cannot find
```

### Functions

```
int func (int a, int& b)   //function header, parameters (each & full)
{
  //function body
  return res;
}
void function: can have return;  return control to calling function
Recursion: Function calling itself; Remember base case
```

```
//We cannot call a function before its definition
//  Solution: Copy the function header at the top (everything);
```

```
Scope of variables: Inside a scope {}, prioritize that one
```

```
Call by value: void f (int a)   Reference: void f(int& a)
Reference cannot be used on a constant (e.g. 2, const int)
```

### Struct

```
struct Point {
   int x,y;
} p1, p2;    name, variable_name are optional
Point pt1={1,2};
p1.x=2; //member access is public by default
//struct do not work with arithmetic and logic (==, <, &&, +-*/)
struct Line {Point p1,p2;};  Line line1={{1,2},{3,4}},line2={1,2,3,4};
```

### Class

```
Class: Expanded concept of a strucutre, Object: An instance of a class
```

```
#include <iostream>
#include “Cpoint.h”
using namespace std;
int main(){
    Cpoint p1,p2;
}
```

```
class Cpoint{
  public: int x, y;
  void set_val
     (int x1, int y1);
  private: int s;
};
```

```
#include <iostream>
#include “Cpoint.h”
using namespace std;
void Clock::set_val
   (int x1,int y1)
     {x=x1; y=y1;}
```

```
//Constructor
Cpoint (int x1, int y1){
  x=x1; y=y1;
}
Cpoint (){  x=0; y=0; }
//Should be public member
Copy:   Cpoint q(p);
Cpoint pt1(1,2); //back
Cpoint pt2; //no () needed
//if no constructr, do nth
```

```
//Destructor
~Cpoint (){
  cout << x << y;
}
//Public member

Friend:
friend int main();
//allow access
private members
```

```
//Operator overload
bool operator ==
  (Cpoint & rhs)
{ return (x==rhs.x);}

//still can access if
x is private member
using the operator
//should be public
+=: return void
```

### Pointers

```
int a, *addr;
addr=&a;
*addr=123;
int c=*addr;
int *y=NULL;
Cpoint *d;
int k=d->x;
```

```
Strictly enforce different address types
int* a, b:    b is int, not int*
int *p        2D: int **v=new int*[y]; v[0]=new int[x]
p = new int; *p=8;        *addr.x   vs  *(addr.x)
delete p;             Name of array is a pointer!
p = new int [10];         p[i]==*(p+i)
FOR (i,0,10) cin >> p[i];     delete []p;
```