

# **C++ LANGUAGE**

## **A Short Course**

**Alvaro F. M. Azevedo**

**<http://www.fe.up.pt/~alvaro>**

# ANSI C++

- Standard (ANSI)
- Compiled - almost as efficient as C
- Object oriented - high level
- Low level coding is still possible
- Other languages are similar to the C++ language
  - ◆ Example: Java, C#, etc.

# MY FIRST FULL PROGRAM

```
#include <iostream.h>

int main()
{
    cout << "Hello world!" << endl;

    return 0;
}
```

♦ Recommended source file extension:  .cpp

Example: `my_first_test.cpp`

# OUTPUT FORMATS

```
// This is a comment
#include <iostream.h> // Required by cout
#include <iomanip.h> // Required by setw, setiosflags, etc.

int i = 901;
cout << "i = " << setw(12) << i << endl;

int j = 902;
cout << "j = " << j << endl; // setw(12) is no longer valid

double f = 2.0 / 3.0;
cout << setiosflags(ios::fixed | ios::showpoint);
cout << setprecision(4);
cout << "f = " << setw(12) << f << endl; // f = 0.6667
```

# INPUT

```
int myAge;

double todaysTemperature;

cout << "How old are you? ";

cin >> myAge; // Read "something" from the keyboard

cout << "Tell me the temperature outside... ";

cin >> todaysTemperature;

cout << "I am " << myAge << " years old and ";

cout << "the temperature" << endl << "outside is ";
cout << todaysTemperature << " degrees." << endl;
```

# CHARACTERS

```
char gender; // Male/Female => 'M'/'F'

cout << "Male (M) or Female (F) ? ";

cin.get(gender); // Reads one character

if (gender != 'M' && gender != 'F')
{
    cerr << "ERROR!" << endl;
    exit(1); // #include <stdlib.h> is required
} // if

cout << "The gender is: " << gender << endl;
```

# STRINGS

```
const int MAX_CHAR = 4;
char t[MAX_CHAR]; // A string is an array of char

cout << "Type a string (max. = 3 characters)... ";
cin.getline(t, MAX_CHAR);

char* pch = t;

while (*pch)
{
    *pch = toupper(*pch); // #include <stdlib.h> is required
    pch++;
}

cout << "The string is now: " << t << endl;
```

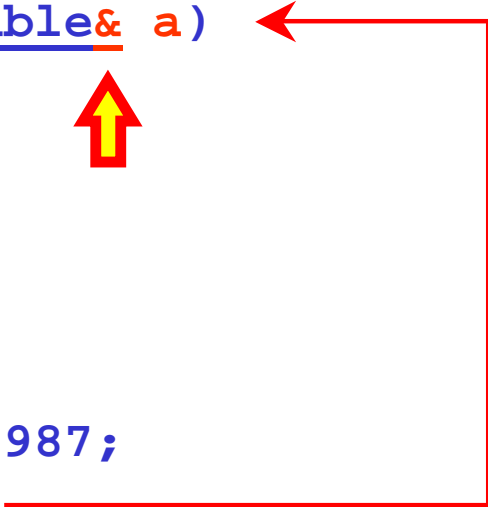
# REFERENCES

```
double zzz = 5.1; // zzz is a double
double& rza = zzz; // rza is a reference to zzz
double& rzb = zzz; // rzb is also a reference to zzz
```

```
cout << "zzz = " << zzz << endl; // zzz = 5.1
cout << "rza = " << rza << endl; // rza = 5.1
cout << "rzb = " << rzb << endl; // rzb = 5.1
cout << "-----" << endl;
rza = 8.6; // zzz, rza and rzb are set = 8.6
cout << "zzz = " << zzz << endl; // zzz = 8.6
cout << "rza = " << rza << endl; // rza = 8.6
cout << "rzb = " << rzb << endl; // rzb = 8.6
cout << "-----" << endl;
rzb = 9.8; // zzz, rza and rzb are set = 9.8
cout << "zzz = " << zzz << endl; // zzz = 9.8
cout << "rza = " << rza << endl; // rza = 9.8
cout << "rzb = " << rzb << endl; // rzb = 9.8
```

# REFERENCES AS FUNCTION ARGUMENTS

```
void f_by_ref(double& a) ←  
{  
    a += 200.0; ↑  
}  
  
int main()  
{  
    double a = 1.987;  
    f_by_ref(a);  
    cout << "a = " << a << endl; // Output: a = 201.987  
  
    return 0;  
}
```



Note: see the file

"by\_copy\_by\_ref\_by\_addr.cpp"

# FILES (input)

```
ifstream f; // #include <fstream.h> is required

f.open("my_file.txt", ios::in | ios::nocreate);
if (f == NULL)
{
    cerr << "File not found" << endl;
    exit(1); // Program terminates
} // if

double z;
f >> z; // The value of z is read in the file

cout << "z = " << z << endl;

f.close(); // The file is closed
```

# FILES (output)

```
ofstream f; // #include <fstream.h> is required
```

```
f.open("my_file.txt", ios::out);  
if (f == NULL)  
{  
    cerr << "Unable to open the output file" << endl;  
    exit(1); // Program terminates  
} // if  
  
double z = -7.15e-3;  
f << z << endl; // The value of z is written in the file  
  
f.close(); // The file is closed
```

# INPUT / OUTPUT FROM A STRING

```
stringstream t; // #include <strstrea.h> is required
double x = -0.00287;
t << x << "654"; // Output goes to the string t
double y;
t >> y; // Input from the string t
cout << setiosflags(ios::scientific);
cout << setprecision(8);
cout << "y = " << setw(16) << y; // y = -2.87654000e-003
cout << endl;
```

# new AND delete

```
int n; cout << "n ? "; cin >> n;

double* vect;
vect = new double [n]; // Allocate memory
if (vect == NULL) { cerr << "No memory!" << endl; exit(1); }

for (int i = 0; i < n; i++)
{
    vect[i] = double(i) / 3; // i is converted into a double
}

for (i = 0; i < n; i++)
{
    cout << i << " / 3 = " << vect[i] << endl;
}

delete [] vect; // Free memory
```

# A SIMPLE class

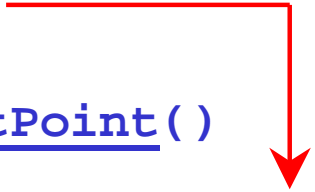
```
class CPoint
{
// Default: private (see what happens when public is removed)
public:
    double x;
    double y;
    int size;
    char color;
}; // Semicolon (;) required

int main()
{
    CPoint A, B; // A and B are instances of the class CPoint
    A.x = 70.1; A.y = 70.2; A.size = 5; A.color = 'R';
    B = A;
    cout << "B = (" << B.x << "," << B.y << ") -> ";
    cout << B.size << "/" << B.color << endl;
    ...
}
```

# A class WITH A MEMBER FUNCTION

```
class CPoint
{
public:
    double x;
    ...
    void PrintPoint()
    {
        cout << "(" << x << "," << y << ") -> ";
        cout << size << "/" << color << endl;
    }
};

int main()
{
    CPoint A, B;
    ...
    A.PrintPoint(); // Prints the content of A
    B.PrintPoint(); // Prints the content of B
    ...
}
```



# DYNAMIC ALLOCATION OF OBJECTS

```
int main()
{
    CPoint* A = new CPoint; // Dynamic allocation of A
    CPoint* B = new CPoint; // Dynamic allocation of B

    A->x = 70.1; A->y = 70.2; A->size = 5; A->color = 'R';
    *B = *A;
    B->color = 'G';

    A->PrintPoint(); // Prints the content of A
    B->PrintPoint(); // Prints the content of B

    delete A; // Free memory
    delete B; // Free memory

    return 0;
}
```

# PRIVATE DATA / PUBLIC FUNCTIONS

```
class CPoint
{
private:
    double x;
    double y;
    ...
public:
    void PrintPoint()
    {
        cout << "(" << x << ...
    }
};
int main()
{
    CPoint A;
    A.x = 70.1; // ERROR: data member cannot be accessed
    A.PrintPoint(); // OK: member function is public
    ...
}
```

Member functions can access private data members.

# DEFAULT CONSTRUCTOR

```
class CPoint
{
private:
    double x;
    ...
public:
    CPoint() // Default constructor
    {
        cout << "### Default constructor called!" << endl;
        x = 0.0;      y = 0.0;
        size = 1;    color = 'W';
    }
};
int main()
{
    CPoint A; // Default constructor is called
    A.PrintPoint(); // Prints the content of A
    ...
}
```

# INITIALIZER CONSTRUCTOR

```
class CPoint
{
private:
    double x;
    ...
public:
    CPoint(double xArg, double yArg, int sizeArg,
           char colorArg) // Initializer constructor
    {
        cout << "### Initializer constructor called!" << endl;
        x = xArg;          y = yArg;
        size = sizeARG;    color = colorArg;
    }
};
int main()
{
    CPoint A(70.1, 70.2, 5, 'R'); // Init. constr. is called
    A.PrintPoint(); // Prints the content of A
}
```

# COPY CONSTRUCTOR

```
class CPoint
{
public:
    CPoint(const CPoint& pointArg) // Copy constructor
    {
        cout << "### Copy constructor called!" << endl;
        x = pointArg.x;          y = pointArg.y;
        size = pointArg.size;    color = pointArg.color;
    }
};

int main()
{
    CPoint A(70.1, 70.2, 5, 'R'); // Init. constr. is called
    CPoint B = A; // Copy constructor is called
    CPoint C(A); // Copy constr. is called (alternate syntax)
    A.PrintPoint(); // Prints the content of A
    B.PrintPoint(); // Prints the content of B
    C.PrintPoint(); // Prints the content of C
}
```