UNIVERSITY of LOUISVILLE
dare to be great

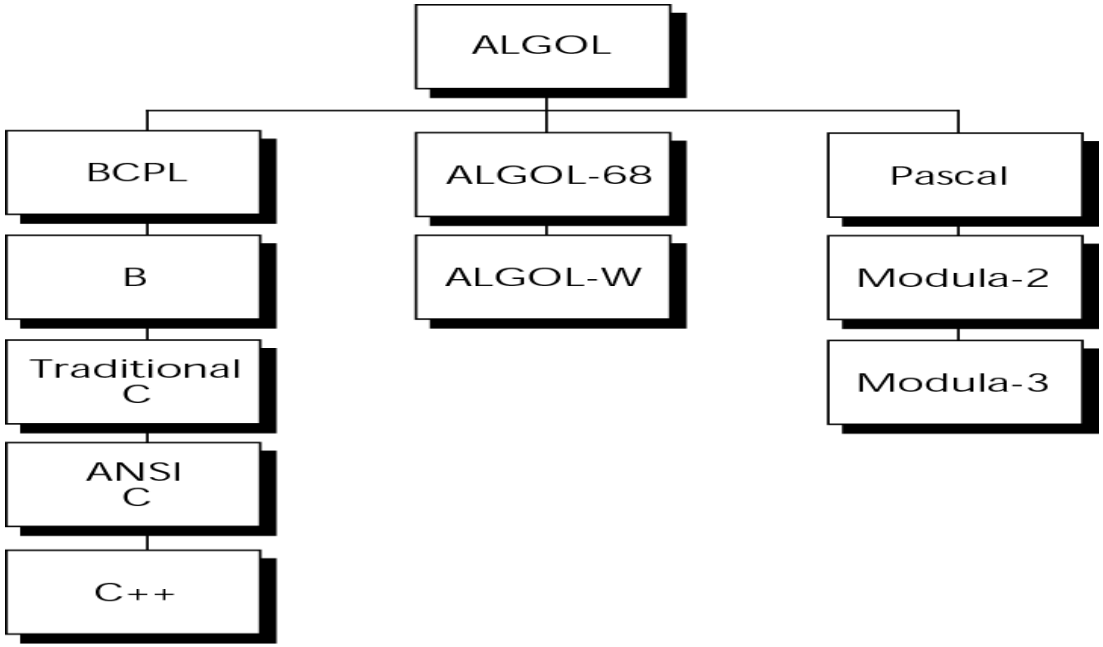
# CPP Chapter 1: Input and Output

CECS130
Introduction to Programming Languages
Dr. Roman V. Yampolskiy

# Introduction to C++

- C and C++ are closely related
- Because you know C you already know 90% of material covered in Ch. 1-4 of your CPP book
- We will concentrate on the 10% you don't know
- From Ch1 we will cover input and output

# History of the C++ language




**Invented in 1979**

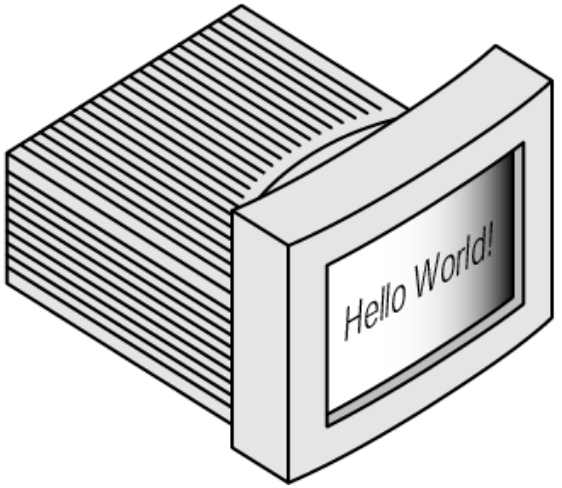
**Invented by Bjarne Stroustrup**

# Hello World program

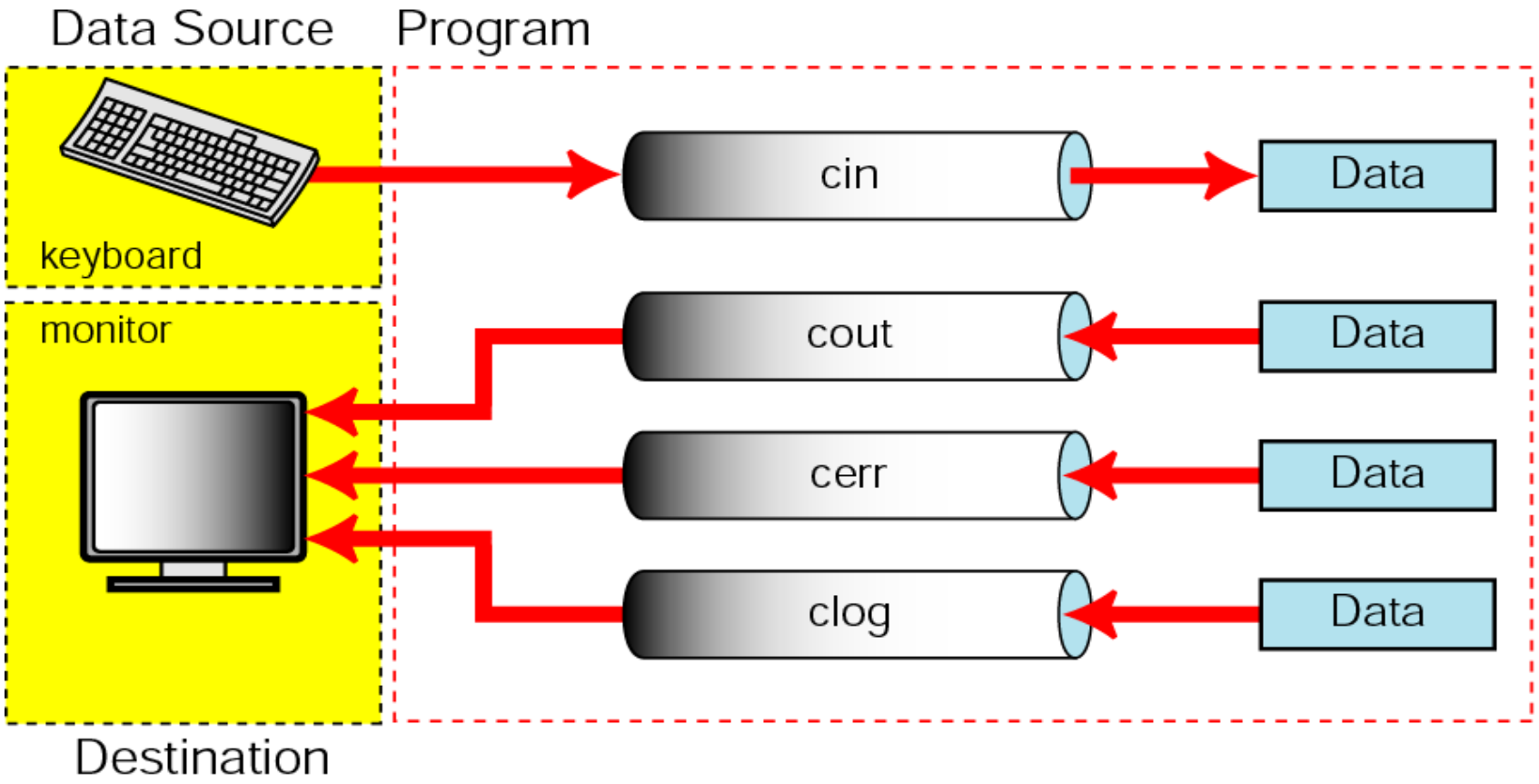
```
#include <iostream>
using namespace std;

int  main  ( )
 {
   cout << "Hello World!\n";
   return  0 ;
 }
```

Preprocessor command to include input/output stream information for your program.



# Standard streams



Note:

***The standard streams are created automatically and connected to appropriate devices when a program starts.***

# cin and cout

You can use the std::cin object to read input from the keyboard and std::cout to write to the monitor.

# C++ Program Example

```
#include <iostream>
int main()
{
  // Display Welcome to C++ to the console
  std::cout << "Welcome to C++!" << std::endl;
  return 0;
}
```

# Omitting the std:: Prefix

std::cout, std::endl, and std::cin all start with std::.

So what is std? std means the standard namespace.

C++ divides the world into “namespaces” to resolve potential naming conflicts.

std::cout means that cout belongs to the standard namespace.

It is tedious to type std:: repeatedly. There is a solution to eliminate the std:: prefix.

Add the statement:

using namespace std;

# Using namespace std

```
#include <iostream>

using namespace std;

int main(void) {

    cout << "Enter a character: ";
    char ch;
    cin >> ch;

    system("pause");
     return 0;
}
```

# Using `cin` and `cout` and namespace

- `cin` and `cout` are declared in the header file iostream, but within a namespace named std
- To use `cin` and `cout` in a program, use the following two statements:

```
#include <iostream>

using namespace std;
```

# Standard I/O Devices

- Use iostream to extract (receive) data from keyboard and send output to the screen
- iostream contains definitions of two types
  - istream - input stream
  - ostream - output stream
- iostream has two variables
  - `cin` - stands for common input
  - `cout` - stands for common output

# `cin` and the Extraction Operator >>

- The syntax of an input statement using `cin` and the extraction operator `>>` is

```
cin >> variable >> variable...;
```

- The extraction operator `>>` is binary
- The left-hand operand is an input stream variable such as `cin`
- The right-hand operand is a variable of a simple data type

# Standard Input

- Every occurrence of `>>` extracts the next data item from the input stream
- Two variables can be read using a single `cin` statement
- No difference between a single `cin` with multiple variables and multiple `cin` statements with one variable
- When scanning, `>>` skips all whitespace
- Whitespace characters consist of blanks and certain nonprintable characters

# Data Type of Input

- `>>` distinguishes between character `2` and number `2` by the right hand operand of `>>`
  - If it is of type `char`, the `2` is treated as character `2`
  - If it is of the type `int` (or `double`) the 2 is treated as the number `2`

# Data Types of Input

Valid Input for a Variable of the Simple Data Type

| Data Type of a | Valid Input for a |
|---|---|
| `char` | One printable character except the blank |
| `int` | An integer, possibly preceded by a + or – sign |
| `double` | A decimal number, possibly preceded by a + or – sign. If the actual data input is an integer, the input is converted to a decimal number with the zero decimal part. |

# Reading Chars

- When reading data into a `char` variable
  - Extraction operator `>>` skips leading whitespace, finds and stores only the next character
  - Reading stops after a single character

# Reading Ints and Doubles

- To read data into an `int` or `double` variable
  - Extraction operator >> skips leading whitespace, reads plus or minus sign (if any), reads the digits (including decimal)
  - Reading stops on whitespace non-digit character

# Examples

```
int a, b;
double z;
char ch, ch1, ch2;
```

| Statement | Input | Value Stored in Memory |
|---|---|---|
| `cin >> ch;` | `A` | `ch = 'A'` |
| `cin >> ch;` | `AB` | `ch = 'A', 'B'` is held for later |
| `cin >> a;` | `48` | `a = 48` |
| `cin >> a;` | `46.35` | `a = 46, .35` is held for later input |
| `cin >> z;` | `74.35` | `z = 74.35` |
| `cin >> z;` | `39` | `z = 39.0` |
| `cin >> z >> a;` | `65.78 38` | `z = 65.78, a = 38` |

# More Examples

```
int a, b;
double z;
char ch, ch1, ch2;
```

| Statement | Input | Value Stored in Memory |
|---|---|---|
| cin >> a >> b; | 4 60 | a = 4, b = 60 |
| cin >> a >> ch >> z; | 57 A 26.9 | a = 57, ch = 'A', z = 26.9 |
| cin >> a >> ch >> z; | 57 A26.9 | a = 57, ch = 'A', z = 26.9 |
| cin >> a >> ch >> z; | 57A26.9 | a = 57, ch = 'A', z = 26.9 |
| cin >> z >> ch >> a; | 36.78B34 | z = 36.78, ch = 'B', a = 34 |
| cin >> z >> ch >> a; | 36.78 B34 | z = 36.78, ch = 'B', a = 34 |
| cin >> a >> b >> z; | 11 34 | a = 11, b = 34, computer waits for the next number |

# Even More Examples

| Statement | Input | Value Stored in Memory |
|---|---|---|
| cin >> a >> z; | 46 32.4 68 | a = 46, z = 32.4,<br>68 is held for later input |
| cin >> a >> z; | 78.49 | a = 78, z = 0.49 |
| cin >> ch >> a; | 256 | ch = '2', a = 56 |
| cin >> a >> ch; | 256 | a = 256, computer waits for the<br>input value for ch |
| cin >> ch1 >> ch2; | A B | ch1 = 'A', ch2 = 'B' |

# Using the `string` Data Type

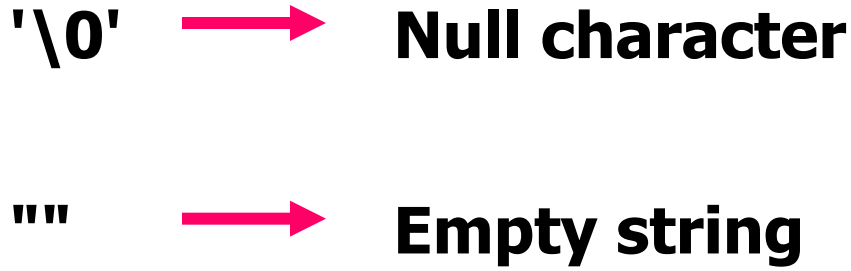
- C++ has a class called **string**
- To use the `string` type, you need to access its definition from the header file `string`
- Include the following preprocessor directive:

```
#include  <string>
```

# Some strings

```
""                                  // A null string
"h"
"Hello World!\n"
"HOW ARE YOU?"
"Good Morning!"
"'Good' Morning!"                   // 'Good' Morning
"\"Good\" Morning!"                 // "Good" Morning
```

# Null characters and null strings



# string/cout example

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

int main(void) {
    string strExample = "Today is Monday";
    cout<<strExample<<endl;
    cout<<"Today is Monday"<<endl;

    system("pause");
    return 0;
}
```

# string/cin example

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

int main(void) {
    string name ="";
    string strPrompt = "What is your name?";
    cout<<strPrompt<<endl;
    cin>>name;
    cout<<name<<endl;

    system("pause");
    return 0;
}
```

# cout example

```
#include <iostream>
using namespace std;
int main()
{
    cout << "My first C++ program." << endl;
    cout << "The sum of 2 and 3 = " << 5 << endl;
    cout << "7 + 8 = " << 7 + 8 << endl;
    return 0;
}
```

**Sample Run**:

```
My first C++ program.
The sum of 2 and 3 = 5
7 + 8 = 15
```

# cout/cin example

```
#include <iostream.h>

int main() {
  int intnum;
  float floatnum;
  double doublenum;
  cout << "Standard prompt: Enter an integer: ";
  cin >> intnum;
  cout << "The value was " << intnum << endl;
  cout << "Enter float: ";
  cin >> floatnum;
  cout << "Result: " << floatnum << endl;
  cout << "Enter double: ";
  cin >> doublenum;
  cout << "Result: " << doublenum << endl << endl;
  cout << "All three values: " << doublenum <<  " " << floatnum << " " << intnum << endl;
  return 0;
}
```

# Bonus Opportunity: Presentation

- This is a course about programming languages.
- Unfortunately we only have time to cover two such languages: C and C++.
- The presentations are aimed to expand the coverage to different programming languages in CECS130.

# Presentation Requirements

- **Up to 5% bonus on your Exam 1 grade**
- **Create and present** (in class) a 15 minute Power Point presentation about a programming language not covered in the course.

# How to proceed?

- You will need to:
  - Pick a programming language
  - A large list of programming languages is available at: http://en.wikipedia.org/wiki/List_of_programming_languages.
- Your presentation needs to cover the following:
  - Introduction
  - History of the language/related languages
  - Design goals/why was the language created
  - Properties/Characteristics/ maybe one or two **short** examples
  - Drawbacks/problems/shortcomings
  - Conclusions/Summary
  - References (at least 10, more is better)

# How to proceed?

- Once you have decided on a language
  - please contact Dr. Yampolskiy to reserve the language you have selected
  - Please tell me what language you have selected by October 31, 2014

# Bonus: Due Dates

- Presentations are due by: 11/25/2014
  - Email me your *.ppt file
  - Be ready to present on 12/1/2014

# The End!