Data Structures & Algorithms - Week 1

Subodh Sharma, Rahul Garg
{svs,rahulgarg}@iitd.ac.in

IIT Delhi, Computer Science Department

1 Course Logistics

2 Introduction to C++

- History
- Variables and Basic Types
- Functions and Selection
- Loops
- Scope - Global and Local
- Scope - Block
- Scope - Static Local Variable
- C++ Pointers
- Array
- Classes

3 Learning C++: In reality

Course Logistics

- All announcement through the course webpage https://subodhvsharma.github.io/course/col106. **So regularly visit and check for updates!**
- All **content-specific discussions on Piazza**. You will be added to COL106's Piazza shortly.
- Course email: **2301-col106@courses.iitd.ac.in**
- Subject header **must** include: **[COL106]**. **Most individual emails will not be entertained**
- **Lab Venue**: LHC 503, LHC 504

Course Logistics

- 6 Quizzes (5), 8 Assignments (20), 2 Lab exams (20)

Course Logistics

- 6 Quizzes (5), 8 Assignments (20), 2 Lab exams (20)
- Mid-sem (25) and major exam (30)

Course Logistics

- 6 Quizzes (5), 8 Assignments (20), 2 Lab exams (20)
- Mid-sem (25) and major exam (30)
- Minimum of 30% in Mid-sem+Final+Quizzes

Course Logistics

- 6 Quizzes (5), 8 Assignments (20), 2 Lab exams (20)
- Mid-sem (25) and major exam (30)
- Minimum of 30% in Mid-sem+Final+Quizzes
- Minimum of 30% in Lab-exams+Lab-assignments

Course Logistics

- 6 Quizzes (5), 8 Assignments (20), 2 Lab exams (20)
- Mid-sem (25) and major exam (30)
- Minimum of 30% in Mid-sem+Final+Quizzes
- Minimum of 30% in Lab-exams+Lab-assignments

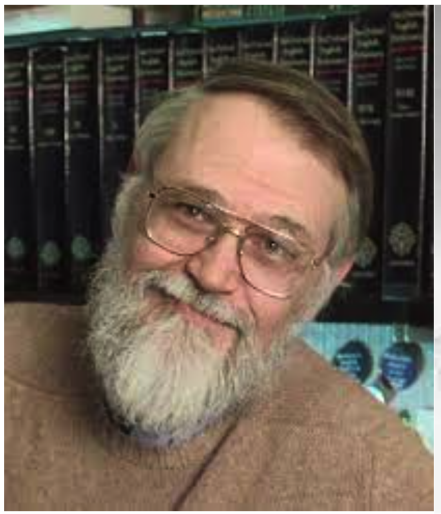
Course Logistics

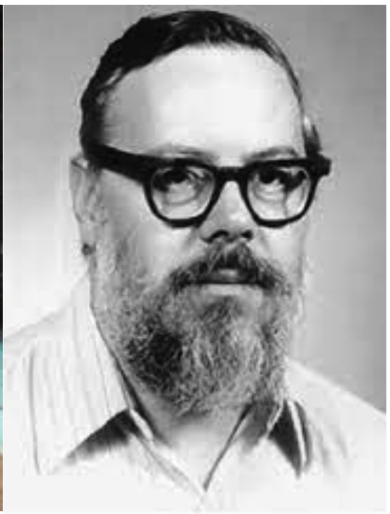
- 6 Quizzes (5), 8 Assignments (20), 2 Lab exams (20)
- Mid-sem (25) and major exam (30)
- Minimum of 30% in Mid-sem+Final+Quizzes
- Minimum of 30% in Lab-exams+Lab-assignments

Book References

- Kernighan, Brian W., and Dennis M. Ritchie. ”The C programming language.” (2002)
- Stroustrup, Bjarne. ”The C++ Programming Language Fourth Edition.” (2013)
- Goodrich, Michael T., Roberto Tamassia, and David M. Mount. Data structures and algorithms in C++. John Wiley & Sons, 2011
- Mark Allen Weiss. Data Structures and Algorithm Analysis in C++. Fourth edition, Pearson.

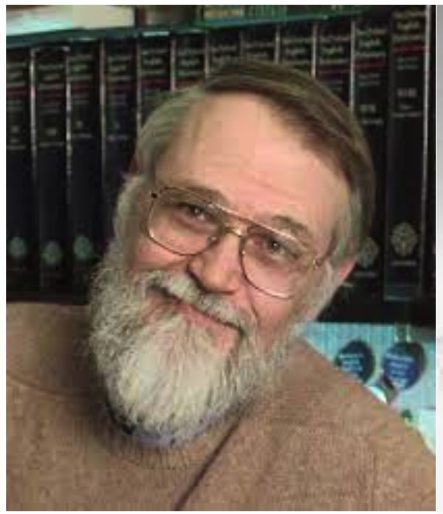
History





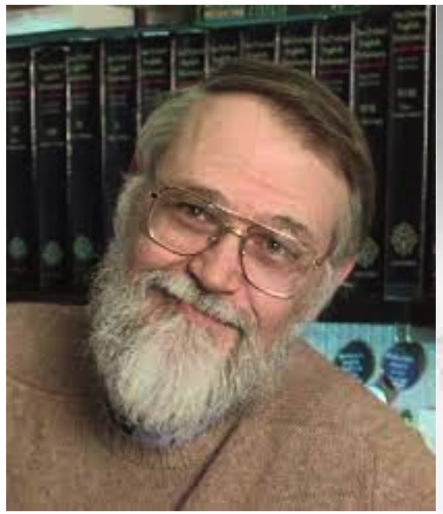
History

- **1978**: Kernighan and Ritchie from Bell Labs published “The C Programming Language”



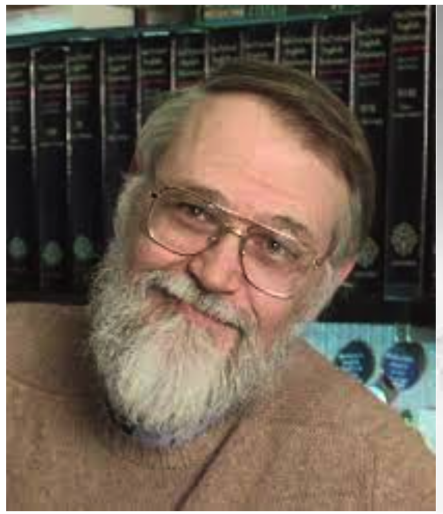
History

- **1978**: Kernighan and Ritchie from Bell Labs published “The C Programming Language”
- **1983**: Barne Stroustrup at Bell Labs introduce C with Classes as C++.



History

- **1978**: Kernighan and Ritchie from Bell Labs published “The C Programming Language”
- **1983**: Barne Stroustrup at Bell Labs introduce C with Classes as C++.
- **2011**: C/C++11 standard introduced with a native support for Concurrency, range-based loops, lambda functions, unique/weak/shared pointers, etc.



Variables and Basic Types

- Variables are symbolic names given to expressions or identifiers

Variables and Basic Types

- Variables are symbolic names given to expressions or identifiers
- Variables store data

Variables and Basic Types

- Variables are symbolic names given to expressions or identifiers
- Variables store data
- Each variable has a type

```
int inch; // declaration
/*
definition
requires declaration first!
*/
inch = 2;
```

Variables and Basic Types

- Variables are symbolic names given to expressions or identifiers
- Variables store data
- Each variable has a type

```
int inch; // declaration
/*
definition
requires declaration first!
*/
inch = 2;
```

- Types determine the operations that can be performed on variables

Variables and Basic Types

- Variables are symbolic names given to expressions or identifiers
- Variables store data
- Each variable has a type

```
int inch; // declaration
/*
definition
requires declaration first!
*/
inch = 2;
```

- Types determine the operations that can be performed on variables
- Basic types: **bool, char, int, float, double**

Functions and Selection


```
// Func declaration
int someFunc(char i);
// Func definition
int someFunc(char input){
  if (input == 'y') return 1;
  else return 0;
}
```

- Function name: `someFunc`
- Function return type: `int`
- Formal parameters: `char input`
- Function declaration: `int someFunc(char input);`

Loops


```
bool accept() {
  int tries = 1;
  while (tries < 4) {
    std::cout << ``Proceed (y/n)?\n'';
    char answer=0;
    std::cin >> answer;
    switch(answer){
      case 'y':
            return true;
      case 'n':
            return false;
      default:
            std::cout >> ``Unknown option\n'';
            tries++; //tries = tries +1
    }
  }
}
```

LoopsContinued


```
#include <iostream>
int main() {
  int sum =0;
  for (int i = 1; i <= 10; i++) {
    sum += i;
  }
  std::cout << sum << std::endl;
return 0;
}
```

Scope - Global and Local


```
int globalVar;  // global variable

void someFunction() {
  globalVar = 10;  // accessible here
  int localVar =5; // not accessible outside
}
```

Scope - Block


```
int main() {
   {    // new block scope starts
      int blockVar = 5;
      // blockVar is accessible here
      {
         // blockVar is accessible here
      }
   }    // block scope ends
   // blockVar is not accessible
   return 0;
}
```

Scope - Static Local Variable


```
void someFnc() {
  static int x = 30; // Static local variable
  x++;
  // ...
}
// x is not accesssible here
// but when someFnc is called again, x = 31
```

Other keywords relating to scope and visibility: **extern, const, public, private, protected, namespace, volatile, mutable**

Home Reading, Try it out in lab sessions!

C++ Pointers

Pointers

In C++, a pointer is a variable that stores the memory address of another variable.

- Declaration:

```
int *p; // p points to an int val
```

C++ Pointers

Pointers

In C++, a pointer is a variable that stores the memory address of another variable.

- Declaration:

```
int *p; // p points to an int val
```

- Definition:

```
int x = 5;
p = &x; // p points to x
```

C++ Pointers

Pointers

In C++, a pointer is a variable that stores the memory address of another variable.

- Declaration:

```
int *p; // p points to an int val
```

- Definition:

```
int x = 5;
p = &x; // p points to x
```

- Dereference:

```
int y = *p;
// y has the same value as x
```

C++ Pointers

Pointers

In C++, a pointer is a variable that stores the memory address of another variable.

- Declaration:

```
int *p; // p points to an int val
```

- Definition:

```
int x = 5;
p = &x; // p points to x
```

- Dereference:

```
int y = *p;
// y has the same value as x
```

- Value modification through pointers:

```
 *p = 20;
// y has the same value as x
```

Array

- Homogeneous container; Contiguous

```
char v[10]; // array of 10 chars
int num[] = {1, 2, 3, 4};
char *p;
p = &v[3];
```

- Arrays are passed to functions as pointers

```
void someFnc(int *num, int size){
  for (int i = 0; i < size; i++){
    std::cout << num[i];
  }
}
```

Reading Assignment:

- Pass by value vs Pass by reference
- Pointers vs Reference
- Use of `const` with pointers
- Unique and Shared pointers

Classes

Classes are user defined datatypes which encapsulated data and functions into a single unit.

```
class Node{
  private:
    int val;
    Node * next;
  public:
    Node(int value, Node *n=nullptr){
      val = value;
      next = n;
    }
};
Node *n1 = new Node(1);
Node *n2 = new Node(2, n1);
```

Classes:Visibity

- **Public,Private, Protected**: Access specifiers that specify the visibility of variables and functions.

Classes:Visibity

- **Public,Private, Protected**: Access specifiers that specify the visibility of variables and functions.
- Public: The entity is accessible from anywhere outside the class definition

Classes:Visibity

- **Public,Private, Protected**: Access specifiers that specify the visibility of variables and functions.
- Public: The entity is accessible from anywhere outside the class definition
- Private: These members are only accessible from within the class itself through member functions or the **friend** functions.

Classes:Visibity

- **Public,Private, Protected**: Access specifiers that specify the visibility of variables and functions.
- Public: The entity is accessible from anywhere outside the class definition
- Private: These members are only accessible from within the class itself through member functions or the **friend** functions.
- Protected: These members are accessible from within the class and its derived classes.

Classes:Example


```
#include <iostream>
#include <string>

class Person {
private:
    std::string aadhaar;

public:
    std::string name;

    void setAadhaar(std::string s) {
        aadhaar = s;
    }

    std::string getAadhaar() const {
        return aadhaar;
    }
};
```

Classes:Inheritance


```
class Base {
public:
virtual  void foo() {
   std::cout << "Base::foo()\n";
}};

class Derived : public Base {
public:
void foo() {
   std::cout << "Derived::foo()\n";
}
void bar() { Base::foo(); }
};
```

Keyword **virtual** is *necessary* for method overriding.

Learning C++: In reality

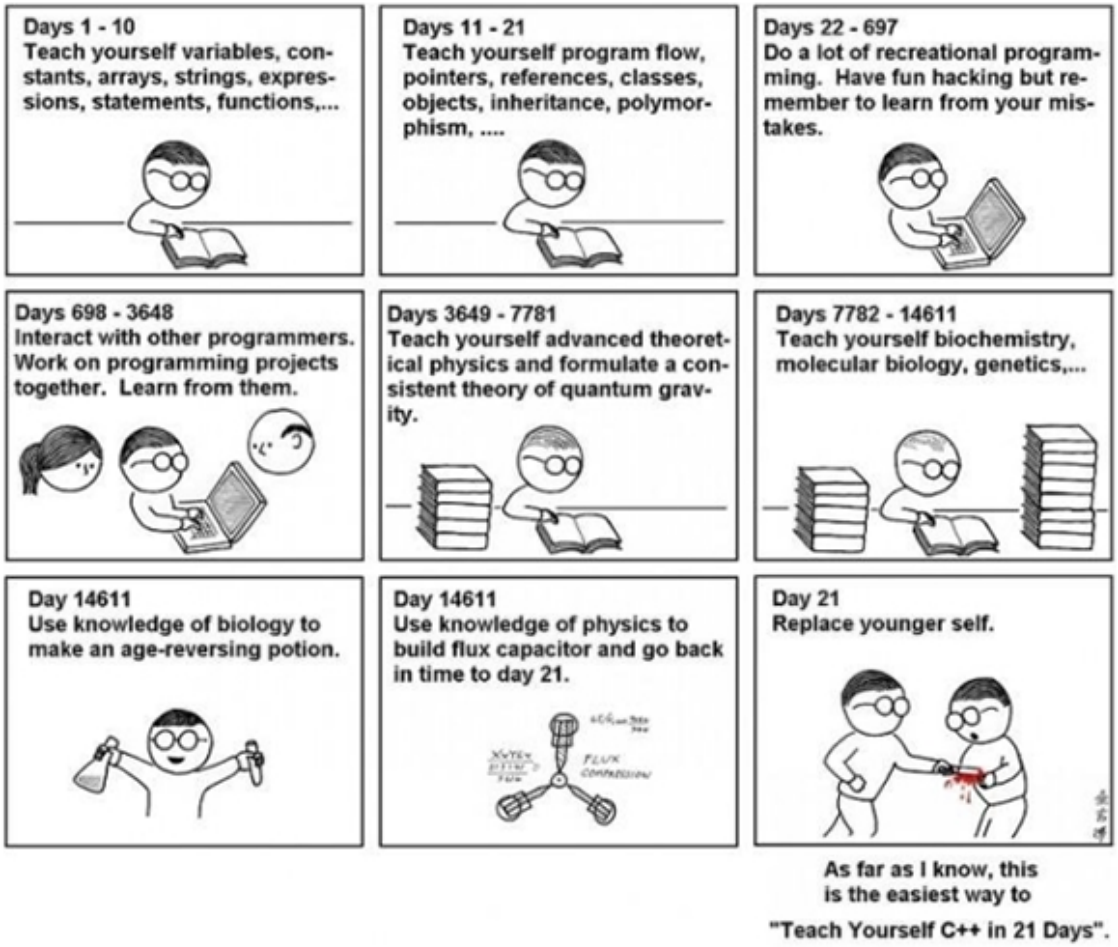


Figure: Learning C++ in 21 days!