

C++ Program Format



- **Compiler sometimes fussy ...**
  - Grammatical rules, syntax
- **...but often doesn't care**
  - Naming conventions, indenting, etc.
- **Human readers important**
  - So "arbitrary" details do matter

---

---

---

---

---

---

---

---

C++ Compiler Flexibility

- **Extra whitespace ignored**
  - Blanks, tabs, whole blank lines, etc.
- **Indentation doesn't matter**
- **Statements per line**
  - More than one statement per line
  - More than one line per statement

---

---

---

---

---

---

---

---

Human Reader Concerns



- **Clean layout**
- **Program structure apparent**
- **Easy to understand intent**
- **Accessible design info**
- **Author identification**

---

---

---

---

---

---

---

---

Text, page 69

Preliminary Style Guidelines (1)

- **One statement per line**
  - Only one data object definition(?)
- **Constant indentation**
  - For code in a single block
- **Separate code sections**
  - Blank lines, comments

---

---

---

---

---

---

---

---

Preliminary Style Guidelines (2)

- **Meaningful identifier names**
  - Consistent naming convention
- **Document with comments**
  - Each identifier
  - Code segment
  - Overall program

---

---

---

---

---

---

---

---

Indenting Blocks (per Textbook)

```

void main ()
{
  int x;
  cin >> x;
  cout << x;
}

```

Two compound statements (blocks)  
But formatted differently!

```

if (discr >= 0)
  root1 = ...x;
  root1 = ...x;
}

```

---

---

---

---

---

---

---

---

### Indenting Blocks (Alternative)

```

void main ()
{
  int x;
  cin >> x;
  cout << x;
}

```

Blocks now formatted in the same way

```

if (discr >= 0)
{
  root1 = ...x;
  root1 = ...x;
}

```

Indent statements inside block by several (2-4?) spaces

This method preferred in this class

---

---

---

---

---

---

---

---

### Program Header

```

// Program for CS-182, lab 3
// Version: 1.0
// Date: 17 September 1997
// Author: Mark Sebern
// Purpose: ...
//
// Libraries used:
#include <iostream>
using namespace std;
...

```

More details may be added later.

---

---

---

---

---

---

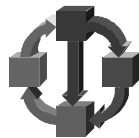
---

---

### Software Development Process

- **Analysis** (requirements)
- **Design** (plan for implementation)
- **Implementation** (code/compile)
- **Testing**

Documentation in all phases




---

---

---

---

---

---

---

---

Analysis



- **Figure out what to do**
  - What is the “client” asking for?
- **Study problem specification**
- **Clarify questionable points**
- **Write your own description**

---

---

---

---

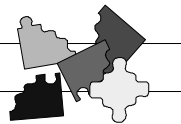
---

---

---

---

Design



- **Identify objects (nouns?)**
  - Some external: problem description
  - Some internal: program segments
- **Identify attributes (adjectives?)**
- **Identify behaviors (verbs?)**

---

---

---

---

---

---

---

---

Implementation



- **Map objects to types**
  - Simple built-in types
  - Complex pre-written types
  - Composites you write
- **Write/review source code**
- **Compile program**

---

---

---

---

---

---

---

---

Testing

- **Identify test cases**

- Sample inputs
- Expected outputs



- **Run program**

- Apply inputs
- Compare outputs
- Modify program as necessary

---

---

---

---

---

---

---

Errors

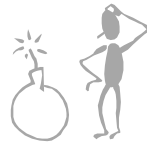
- **Syntax errors**

Text, pages 70-73

- Caught by compiler (maybe not!)
- Error versus warning

- **Runtime errors**

- E.g., divide by zero value



- **Logic errors**

---

---

---

---

---

---

---