

CS120 – Computer Science I

Instructor: Jia Song

Instructor Contact Information

Instructor: Dr. Jia Song

Email: jsong@uidaho.edu (Preferred)

Phone: (208) 885-1710

Office: JEB 240 (CSDS Security Lab)
JEB 340

Office Hour:

MTW 2:30pm – 3:30pm

Schedule an appointment by Email

- Use “**CS120** YOUR SUBJECT HERE” in the subject field.
- Look for a response email (in 24 hours) if you send documents to me.

About this course

- **Time and Location:**

Lecture: WMF 12:30pm – 1:20 pm (MCCL 209)

Labs: Section 1 – T 10:30am – 12:20pm (JEB 321)

Section 2 – T 12:30pm – 2:20pm (JEB 321)

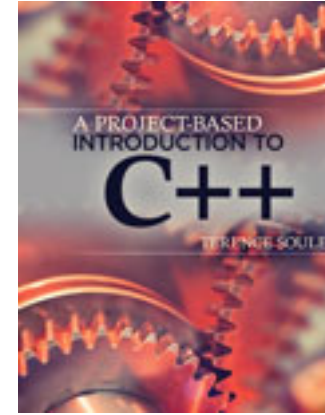
Section 3 – R 2:30pm – 4:20pm (JEB 321)

Learning Outcomes

After successfully completing this course, each student will have learned how to do the following:

- Apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline (ABET student outcome A (introduce))
- Design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs (ABET student outcome C (introduce))
- Function effectively on teams to accomplish a common goal (ABET student outcome D (introduce))
- Communicate effectively with a range of audiences (ABET student outcome F (introduce))
- Use current techniques, skills, and tools necessary for computing practice (ABET student outcome I (introduce))
- Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates an understanding of the tradeoffs involved in design choices (ABET student outcome J (introduce))
- Apply design and development principles in the construction of software systems of varying complexity (ABET student outcome K (introduce))

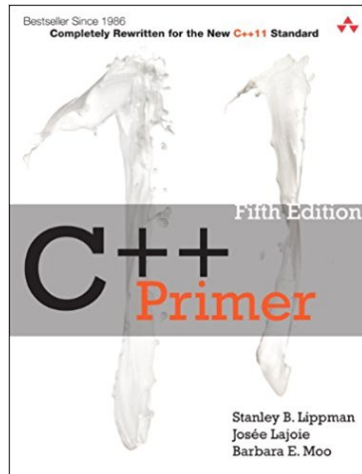
Textbook



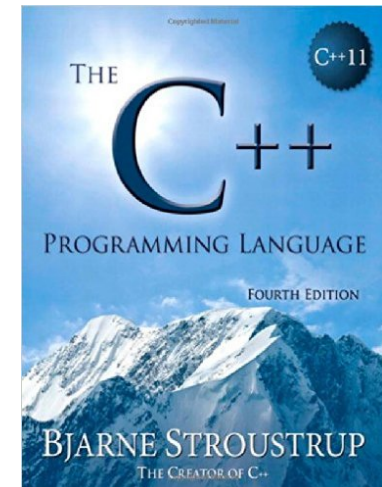
- A Project Based Introduction to C++, Terence Soule, Kendall-Hunt, 2014 (ISBN 9781465251145)
- <http://www.kendallhunt.com/store-product.aspx?id=268763>

Recommended C++ books

- C++ Primer (Stanley B. Lippman, Josée Lajoie, Barbara E. Moo)

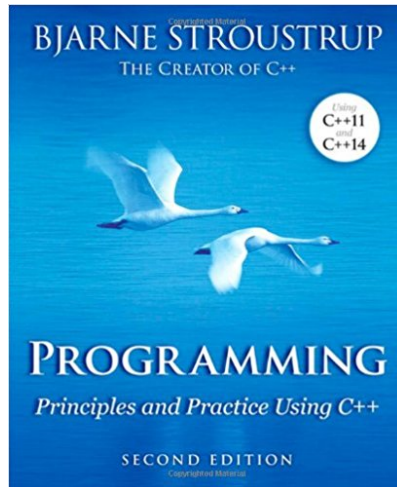


- The C++ Programming Language (Bjarne Stroustrup)

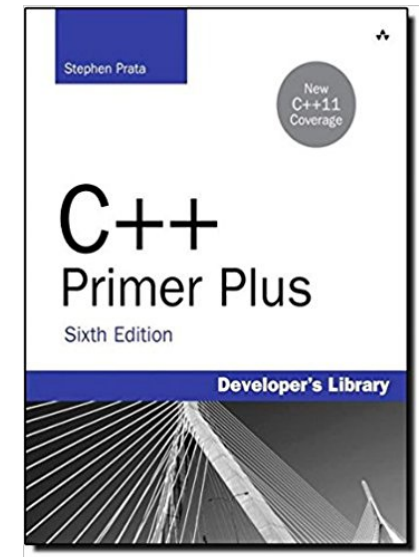


Recommended C++ books

- Programming: Principles and Practice Using C++ (Bjarne Stroustrup)



- C++ Primer Plus (Stephen Prata)



Course website

- <http://www2.cs.uidaho.edu/~jsong/cs120spring2018.html>

Course Schedule

Week	Date	Topics	Chapters
1	Jan 10 – Jan 12	Introduction to Computers and programming	1
2	Jan 15 – Jan 19	Variables, basic I/O, conditionals, libraries ***Idaho Human Rights Day Jan 15 (No classes)***	2
3	Jan 22 – Jan 26	Conditionals, loops, random numbers	3
4	Jan 29 – Feb 2	More Loops	3
5	Feb 5 – Feb 9	Functions	4
6	Feb 12 – Feb 16	Classes ***short exam #1 Fri Feb 16***	5
7	Feb 19 – Feb 23	More classes ***Presidents Day Feb 19 (No classes)***	5

Course Schedule

8	Feb 26 – Mar 2	Software design and engineering	5
9	Mar 5 – Mar 9	Passing arrays to functions, binary, hex	6
10	Mar 12 – Mar 16	***Spring break – No classes***	
11	Mar 19 – Mar 23	Arrays	6
12	Mar 26 – Mar 30	Two-dimensional arrays ***short exam #2 Fri Mar 30***	7
13	Apr 2 – Apr 6	Pointers	7
14	Apr 9 – Apr 13	More pointers	7
15	Apr 16 – Apr 20	Linked lists	8
16	Apr 23 – Apr 27	Recursion	8
17	Apr 30 – May 4	Final Exam Review	
18	May 7 – May 11	Final Exam	

Grading

Two 50-minutes exams	20%
Final exam	20%
Assignments	30%
Quizzes	15%
Labs	15%

Score	Grade
90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
0 - 59	F

How to learn a programming language?

- Practice, practice and practice! (Homework, Lab)
- Review what you have learned before! (Quiz, exam)
- Ask for help when you need! Don't wait!
 - Ask me
 - Ask TA in the lab
 - Stop by CSAC (Computer Science Assistance Center) located in JEB211D

Terms and Conditions

You are responsible for:

- Reading and Understanding the course material.
- Turning in your homework, lab reports and other assignments **ON TIME**.
Weekly homework will be due **every Sunday 11:59pm**.
(Late homework: 1 day – 70%, 2 days – 50%, more than 2 days – 0)
- Attending the class and pay attention in class.
- Finding out the exam times (**2 short exams and 1 final exam**) and being there.
- No cheating! Cheating will result in failing the class.

Terms and Conditions

- Group work

Homework and labs can be conducted in groups. I encourage working in group and sharing the knowledge. However, **you must understand the materials and turn in your individual copy of homework and lab report.**

- No group work on exams.

- Questions?

Programming

- **Programming** is the process of writing a program that a computer can run.
- A **programming language** is a specific set of basic instructions that can be combined to create a program.

Hello world! in different languages

- Assembly language

```
global _main
extern _printf

section .text
_main:
    push    message
    call    _printf
    add     esp, 4
    ret
message:
    db     'Hello, World', 10, 0
```

- BASH (UNIX SHELL)

```
#!/bin/bash
STR="Hello World!"
echo $STR
```

- Applescript

```
say "Hello, world!"
```

From: <https://excelwithbusiness.com/blog/say-hello-world-in-28-different-programming-languages/>

Hello world! in different languages

- C

```
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

- C++

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
    return 0;
}
```

- FORTRAN

```
program helloworld
    print *, "Hello world!"
end program helloworld
```

- JAVA

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Hello world! in different languages

- MACHINE CODE

```
b8 21 0a 00 00 #moving "!\\n" into eax
a3 0c 10 00 06 #moving eax into first memory location
b8 6f 72 6c 64 #moving "orld" into eax
a3 08 10 00 06 #moving eax into next memory location
b8 6f 2c 20 57 #moving "o, W" into eax
a3 04 10 00 06 #moving eax into next memory location
b8 48 65 6c 6c #moving "Hell" into eax
a3 00 10 00 06 #moving eax into next memory location
b9 00 10 00 06 #moving pointer to start of memory location into
ecx
ba 10 00 00 00 #moving string size into edx
bb 01 00 00 00 #moving "stdout" number to ebx
b8 04 00 00 00 #moving "print out" syscall number to eax
cd 80 #calling the linux kernel to execute our print to
stdout
b8 01 00 00 00 #moving "sys_exit" call number to eax
cd 80 #executing it via linux sys_call
```

High-level languages vs Low-level languages

- High-level languages:
 - human readable
 - hard for computer to understand
 - eg: Java, C, C++, Python, ...
- Low-level languages:
 - hard for people to understand
 - easy for computers to understand
 - eg: Assembly language

Levels of Programming Languages

High-level program

```
class Triangle {  
    ...  
    float surface()  
        return b*h/2;  
}
```

Low-level program

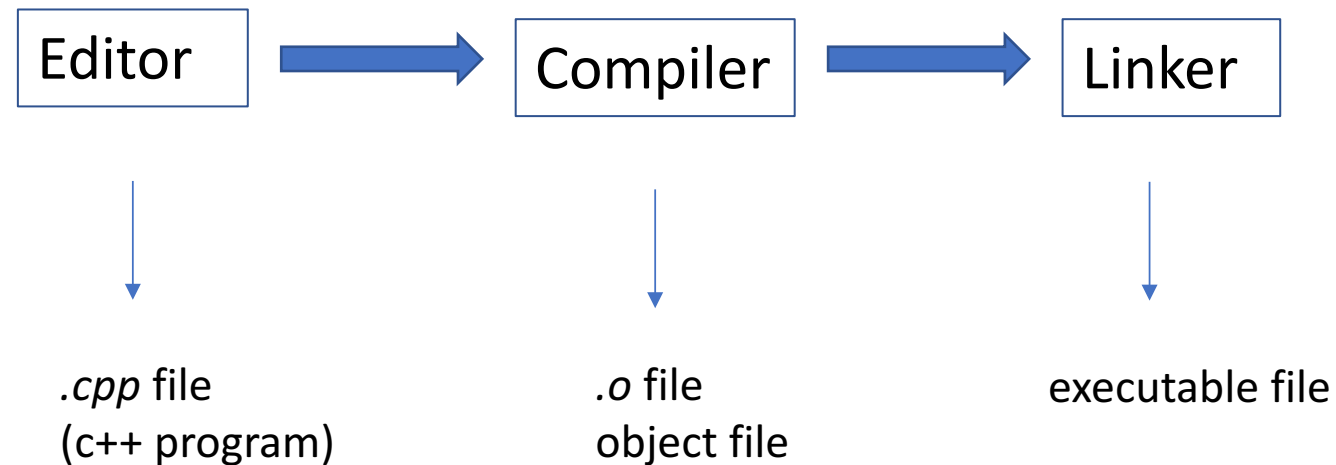
```
LOAD r1,b  
LOAD r2,h  
MUL r1,r2  
DIV r1,#2  
RET
```

Executable Machine code

```
0001001001000101  
0010010011101100  
10101101001...
```

C++ Environment

- Because computers cannot directly execute a program written in a high-level language... -- **Compiler & interpreter**



- **Compiler** – translates the entire program
- **Interpreter** – translates a program one line at a time, as the computer is running the program.

IDE (Integrated Development Environment)

- GNU nano
 - <https://www.nano-editor.org/>

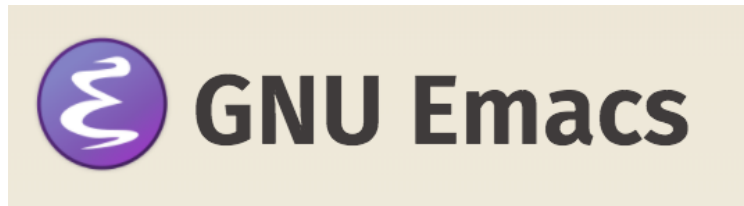
```

                                     :::
                                     The
iLE88Dj. :jD88888Dj:
.LGitE888D.f8GjjjL8888E;
iE :8888Et. .G8888.
;i E888, ,8888,
D888, :8888:
D888, :8888:
D888, :8888:
D888, :8888:
888W, :8888:
W88W, :8888:
W88W: :8888:
DGGD: :8888:
                                     .d8888b. 888b 888 888 888
d88P Y88b 8888b 888 888 888
888 888 88888b 888 888 888
888 888Y88b 888 888 888
888 88888 888 Y88b888 888 888
888 888 888 Y88888 888 888
Y88b d88P 888 Y8888 Y88b. .d88P
"Y8888P88 888 Y888 "Y88888P"
                                     88888b. 8888b. 88888b. .d88b.
888 "88b "88b 888 "88b d88"88b
888 888 .d888888 888 888 888 888
888 888 888 888 888 888 Y88..88P
888 888 "Y888888 888 888 "Y88P"
E888i
tW88D
                                     Text Editor
```

```
GNU nano 2.0.9      File: txt_files/testfile      Modified
Learn how to use nano to boost your terminal confidence!
Edit config files like a pro!
Make easy to-do lists and notes in a text-only format!
Do it via SSH from a smartphone or other computer!
# /etc/fstab: static file system information.
#
# Use 'blkid -o value -s UUID' to print the universally unique identifier
# for a device; this may be used with UUID= as a more robust way to name
# devices that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
# / was on /dev/sdb1 during installation
[ Read 17 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

IDE (Integrated Development Environment)

- <https://www.gnu.org/software/emacs/emacs.html>

A screenshot of the Emacs editor window titled 'emacs@blueberry'. The window has a menu bar with 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Emacs-Lisp', and 'Help'. Below the menu bar is a toolbar with icons for file operations and editing. The main text area contains Lisp code defining functions for hash-table operations:

```
(defsubst hash-table-empty-p (hash-table)
  "Check whether HASH-TABLE is empty (has 0 elements)."
  (zerop (hash-table-count hash-table)))

(defsubst hash-table-keys (hash-table)
  "Return a list of keys in HASH-TABLE."
  (let ((keys '()))
    (maphash (lambda (k _v) (push k keys)) hash-table)
    keys))

(defsubst hash-table-values (hash-table)
  "Return a list of values in HASH-TABLE."
  (let ((values '()))
    ...))
```

 Below the code, the status bar shows '--- subr-x.el.gz 36% L148 (Emacs-Lisp)'. The bottom pane displays the documentation for the 'emac' command, titled 'The Emacs Editor', which describes Emacs as an extensible, customizable, self-documenting real-time display editor. The bottom status bar shows 'U:%%- *info* (emacs) Top Top L9 (Info Narrow)'.

- Assignment #0 (Do not need to turn in)
 - Set up the environment
 - login to cs server (wormulon.cs.uidaho.edu)
 - type, compile and run your first c++ program
- No class on Monday Jan 15
- Will have labs next week

History of C and C++

History of C:

- C is a programming language developed in the 1970's alongside the UNIX operating system.
- Evolved from two other programming languages
 - BCPL and B
- Dennis Ritchie (Bell Laboratories)
 - Added data typing, other features
- C provides a comprehensive set of features for handling a wide variety of applications, such as systems development and scientific computation.

History of C and C++

History of C++:

- Extension of C
- Early 1980s: Bjarne Stroustrup (Bell Laboratories)
- Provides capabilities for object-oriented programming
 - Objects: reusable software components
 - Model items in real world
 - Object-oriented programs
 - Easy to understand, correct and modify

Vedio: Bjarne Stroustrup: Why I Created C++

<https://www.youtube.com/watch?v=JBjjnqG0BP8>