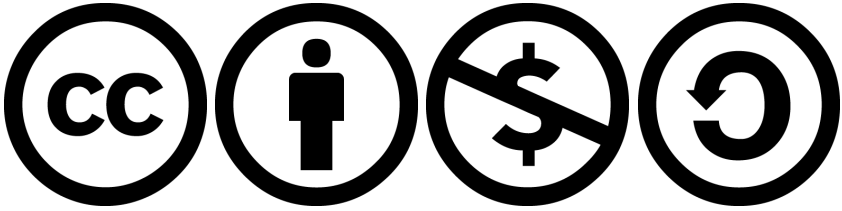
The Rook's Guide to C++

Kickstarter Backer & Contributor Version

August 1, 2013

Contents

License



Dramatis Personæ

Managing Editor:

Jeremy A. Hansen, PhD, CISSP

Technical Editing & Typesetting:

Jeremy A. Hansen

Matt Jadud, PhD

Craig D. Robbins

Theodore M. Rolle, Jr.

Media & Outreach:

Matthew E. Russo

Cover Art & Graphic Design:

Allyson E. LeFebvre

Content Authors:

Tyler Atkinson, Troy M. Dundas, Connor J. Fortune, Jeremy A. Hansen, Scott T. Heimann, Benjamin J. Jones, Michelle Kellerman, Michael E. Kirl, Zachary LeBlanc, Allyson E. LeFebvre, Gerard O. McEleney, Phung P. Pham, Megan Rioux, Alex Robinson, Kyle R. Robinson-O'Brien, Jesse A. Rodimon, Matthew E. Russo, Yosary Silvestre, Dale R. Stevens, Ryan S. Sutherland, James M. Verderico, Christian J. Vergnes, Rebecca Weaver, Richard Z. Wells, and Branden M. Wilson.

Funding & Support:

Peter Stephenson, PhD, VSM, CISSP, CISM, FICAF, LPI at the Norwich University Center for Advanced Computing & Digital Forensics

Andrew Pedley at Depot Square Pizza

Kickstarter contributors:

Nathan Adams, Chris Aldrich, Jay Anderson, Kent Archie, Erik Arvedson, Astrolox, Phoebe Ayers, Papa Joe Barr, Julia Benson-Slaughter, Georgia Perimeter College, Patrick Berthon, Francis Bolduc, Greg Borenstein, Mark Braun, Patrick Breen, Igor Bronshteyn, Valdemar Bučilko, Ross Buckley, Nikita Burtsev, Jakob Bysewski, David Camara, Dave M. Campbell, Brian V. Campbell III, S. Canero, Serge Canizares, Andrew Carlberg, Casey B. Cessnun, Winston Chow, W. Jesse Clements, Greg Crawford, Sean Cristofori, Jordan G Cundiff, Michael David, Joseph Davies, Ashley Davis, David C. Dean, DJS, Carlton Doc Dodd, Phil Dodds, Dominic, Sankar Dorai, dryack, Matt DuHarte, Brandon Duong, Van Van Duong, Daniel Egger, Chris Fabian, Jorge F. Falcon, Tek Francis, Fuchsi, Steve Gannon, Michael Gaskins, Gavlig, Adam Gibson, Russell E. Gibson, Goldenwyrm, James Green, Brian J. Green, Casey Grooms, Vitalik Gruntkovskiy, Vegar Guldal, Felix Gutbrodt, Jeremy Gwinnup, Beau T. Hahn, Paul R. Harms - Norwich 1975, Corey H. Hart, MBA, Aaron A. Haviland, Josh Heffner, Greg Holland, Henry Howard, Mark V Howe, Ivaliy Ivanov, Matt Jadud, Joseph Jaoudi, Tim R. Johnson, Ibi-Wan Kentobi, Mark King, Mitchell Kogut III, Sigmund Kopperud, Michael Korolev, Jamie Kosoy, Aria Kraft, Alexander Týr Kristjánsson, Richard Kutscher, Eric Laberge, John Lagerquist, Philip Lawrence, Mark Brent Lee, John and Nancy LeFebvre, Nevin :-) Liber, Jonathan Lindquist, Thomas Lockney, Stuart A. MacGillivray, Dr. Pedro Maciel, Troels Holstein Madsen, William Marone, Fred Mastropasqua, Miles Mawyer, michael mazzello, Ryan McDonough, Matthew McFadden, John McIntosh II, Sean McNamara, mdsaravanan, Brandon Meskimen, Andrew Mike, G.F. Miller IV, Marcus Millin, Salvador Torres Morales, Danny Morgan, Ken Moulton, Aaron Murray, mvi, Jon Nebenfuhr, Philip K. Nicholson, chris nielsen, Pontus Nilsson, Mike Noble, Aleksander R. Nordgarden-Rødner, Greg O'Hanlon, Doug Otto, Randy Padawer, Ph.D., J Palmer, Tasos Papastylianou, Paul, James Pearson-Kirk, Matthew Peterson, Grigory Petrov, pezmanlou, Joachim Pileborg, Kyle Pinches, pkh, Mary Purdey, Marshall Reeves, Matthew Ringman, Craig D. Robbins, Antonio Rodriguez, Armando Emanuel Roggio, Victor Suarez Rovere, Christian Sandberg, Jaymes Sattler, Paolo Scalia, Patrice Scheidt, Daniel Schmitt, Levi Schuck, Raman Sharma_Himachali, Michael Shashoua, Daniel Shiffman, Clay Shirky, sillygoatgirl, Kevin J. Slonka, Brian Smith, Hazel Smith & Rebecca Twigg, Andrey Soplevenko, Kasper Souren, Derek A. Spangler, Speckman, Kellan St.Louis, Nick Stefanou, Steve, Andrew Stewart, Jeremy Sturdivant, Cyrille Tabary, Adam 8T Tannir, M Taylor, Telecat Productions, Aron Temkin, Mitchell Tilbrook, Nathan Tolbert, Devin M. Tomlinson - Vermont Born, Todd Trimble, Michiel van Slobbe, James A. Velez, Marco Verdecchia, David Walter, Lothar Werzinger, Wayne West, Sean Whaley '05 & M'08, Mark Wheeler, Tommy Widenflycht, Dylan Widis, Tony Williami-

tis, Adam M. Williams, Stephen D. Williams, Dylan Wilson, Wesley Wiser, wizzy, Sam Wright, Janet Hong Yam, and Jy Yaworski.

Chapter 1

History

Chapter 2

Sample Program

Chapter 3

Variables

Chapter 4

Constants

Chapter 5

Assignments

Assignments are a way for a user or a programmer to assign a value to a variable. The way we assign a value to a variable in C++ is different from how we might do it in math. In mathematics we are allowed to say that x = 3 or 3 = x, but in C++ the only acceptable way to assign the value of 3 to x is to type x = 3.

The = in the expression x = 3 is known as an **assignment operator**. This allows the program to set a variable's value depending on its type. Here are some examples of setting a value to different types of variables:

```
int x = 4;
```

```
char alpha = 'A';
```

```
string word = "Alpha";
```

```
float y = 3.14;
```

We are able to declare variables and assign a value to those variables immediately by using the assignment operator. When we assign literal values to variables

of type char, the value must be surrounded by single quotes (for example, `'A'`). When we assign values to variables of type `string`, the literal value must be surrounded by double quotes (for example, `"Alpha"`). We do not have to initialize the values of the variables, however. We can set them later in the code like this:

```
int myVal;
//some code
myVal = 0;
```

In all of the lines of code in this section where a variable is set using the assignment operator, the "thing that is being given a value" is known as an **lvalue**, and the expression on the right that is being stored in the variable is known as the **rvalue**. Literals such as `'A'` or 3 can never be an lvalue. Aside from literals, the rvalue can consist of other variables, like this:

```
myVal = myVal2;
```

Even though `myVal2` is a variable, we are only using the *value stored in the variable*, not the variable itself. For example, if `myVal2` had a value of 6, `myVal` would then be assigned to the value 6 with the above code.

We can also store the results of an arithmetic expression in a variable like this:

```
myVal = 5 + 6; //assigns myVal a value of 11
```

But we can't write

```
5 + 6 = myVal; // ERROR!
```

since `5 + 6` doesn't refer to a place where we can store a value. We can also combine arithmetic expressions with variables as an rvalue like this:

```
myVal2 = 6;
myVal = 4 + myVal2;
```

In this case, the variable `myVal` would be assigned a value of `10` because the variable `myVal2` was initialized to a value of 6, and `4 + 6` is `10`. The value

of `myVal2` remains unchanged. Make sure that the variable `myVal`, the variable `myVal2`, and the literal `4` are of the same type. For example, the following code will result in an error:

```
  int myValue = 4;
  int yourVal;
  string myString = "word";

  yourVal = myValue + myString;
  // Adding string to an int is
  // probably not what you meant!
\begin{lstlisting}

When we try to combine different variable types, the
    compiler will get very mad at us.
Some exceptions to this rule are if we try to combine \Code{
    float}s, \Code{int}s, and \Code{double}s.
These types have the ability to be combined (to a certain
    extent) because they are all numeric values.
Both \Code{double}s and \Code{float}s can hold values with a
     decimal point such as \Code{-3.14}, \Code{0.003}, or \
    Code{5.167289} whereas an \Code{int} can only hold round
     values such as \Code{2}, \Code{-18}, or \Code{100}.
    Refer to Chapter \ref{chap_datatypes} for more
    information on converting between data types.



\LevelD{Review Questions}

\LevelD{Homework Questions}

\LevelD{Review Answers}

\LevelD{Homework Answers}

\LevelD{Further Reading}

\begin{itemize}
\item ~
\item ~
\item ~
\end{itemize}


     \LevelC{Input}
      \input{chap_input.tex}
```

```
      \LevelC{Output}]
       \input{chap_output.tex}
      \LevelC{Arithmetic}
       \input{chap_arithmetic.tex}
      \LevelC{Comments}
       \input{chap_comments.tex}
      \LevelC{Data types, conversion}
       \input{chap_datatypes.tex}

% \LevelA{Section 3}
%   \LevelB{Chapters:}
      \LevelC{Conditionals}
       \input{chap_conditionals.tex}
      \LevelC{Strings}
       \input{chap_strings.tex}
      \LevelC{Loops}
       \label{chap_loops}
       \input{chap_loops.tex}
      \LevelC{Arrays}
       \input{chap_arrays.tex}
      \LevelC{Blocks, Functions, and Scope}
       \input{chap_functions.tex}
      \LevelC{Problem Solving}
       \input{chap_problems.tex}
      \LevelC{Testing}
       \input{chap_testing.tex}

% \LevelA{Section 4}
%   \LevelB{Chapters}
      \LevelC{Preprocessor}
       \input{chap_preproc.tex}
      \LevelC{Advanced Arithmetic}
       \input{chap_advancedarith.tex}
      \LevelC{File I/O}
       \input{chap_file_io.tex}
      \LevelC{Pointers}
       \input{chap_pointers.tex}
      \LevelC{Dynamic Data}
       \input{chap_dynamic.tex}

  \Comment{ % LevelX comment.

  \LevelA{}
    \LevelB{}
      \LevelC{}
      \LevelC{}
      \LevelC{}
```