Introduction to
Computer Programming

With C++

Solutions to Exercises

Spring 2014

Alexis Maciel
Department of Computer Science
Clarkson University

Copyright © 2014 Alexis Maciel
Contents

Preface v

1 Basic Concepts 1
  1.1 A Running Pace Calculator 1
  1.2 High-Level Languages and Compilers 1
  1.3 Output and Data 1
  1.4 Structure of a Simple C++ Program 2
  1.5 Input and Variables 2
  1.6 Arithmetic Operations 3
  1.7 More on Variables 4
  1.8 Conditional Statements 5

2 Repetition 9
  2.1 Adding a Loop to the Running Pace Calculator 10
  2.2 More on Loops 11
  2.3 Loops and Variables 12
  2.4 Nested Loops 13

3 File Input and Output 15
  3.1 A Pay Calculator 15
  3.2 File Streams 16
  3.3 Detecting the End of the File 18
  3.4 Different Wages 23
  3.5 String Variables 23
  3.6 Error Checking 25
  3.7 Extending the Pay Calculator 25
  3.8 Compile-Time Constants 28
  3.9 Formatting of Floating-Point Numbers 28
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Functions</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>A Rounding Function</td>
<td>29</td>
</tr>
<tr>
<td>4.3</td>
<td>Functions in the Pay Calculator</td>
<td>30</td>
</tr>
<tr>
<td>4.4</td>
<td>Reference Arguments</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>Reference Arguments in the Pay Calculator</td>
<td>31</td>
</tr>
<tr>
<td>4.6</td>
<td>Modularity and Abstraction</td>
<td>31</td>
</tr>
<tr>
<td>4.7</td>
<td>Documentation</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>Vectors</td>
<td>33</td>
</tr>
<tr>
<td>5.1</td>
<td>A Simple File Viewer</td>
<td>33</td>
</tr>
<tr>
<td>5.2</td>
<td>Vector Basics</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Object-Oriented Programming</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Design and Implementation of the File Viewer</td>
<td>35</td>
</tr>
<tr>
<td>5.5</td>
<td>Multiway Branches</td>
<td>35</td>
</tr>
<tr>
<td>5.6</td>
<td>More on Vectors</td>
<td>36</td>
</tr>
<tr>
<td>5.7</td>
<td>More on Strings</td>
<td>39</td>
</tr>
<tr>
<td>5.8</td>
<td>A Simple Text Editor</td>
<td>39</td>
</tr>
<tr>
<td>5.9</td>
<td>Adding More Error-Checking to the File Viewer</td>
<td>39</td>
</tr>
<tr>
<td>5.10</td>
<td>Arrays</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>Structures</td>
<td>41</td>
</tr>
<tr>
<td>6.1</td>
<td>Extending the Pay Calculator</td>
<td>41</td>
</tr>
<tr>
<td>6.2</td>
<td>Improving the Design of the Text Editor</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>Algorithms and Generic Programming</td>
<td>47</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>47</td>
</tr>
<tr>
<td>7.2</td>
<td>Generic Programming</td>
<td>47</td>
</tr>
<tr>
<td>7.3</td>
<td>Some Simple Algorithms</td>
<td>48</td>
</tr>
<tr>
<td>7.4</td>
<td>Algorithms in the STL</td>
<td>50</td>
</tr>
</tbody>
</table>
Preface

This document contains solutions to many of the exercises of the course notes Introduction to Computer Programming: With C++. These notes were written for the course CS141 Introduction to Computer Science I taught at Clarkson University. The solutions are organized according to the same chapters and sections as the notes.

Whether you are taking a class or going through these notes on your own, your goal should be to understand the material well enough that you can do the exercises yourself. Simply studying the solutions is not the best way to achieve this. It is much better to spend a reasonable amount of time and effort trying to do the exercises yourself before you look at the solutions.

When an exercise asks you to write C++ code, you should compile your code and test it. In the real world, programmers don’t have solutions against which to check their code. Testing is a critical tool for ensuring that code is correct, so you need practice it.

Now, if you can’t get a solution to an exercise on your own, you should study the notes some more. If that doesn’t work, ask for help from another student or from your instructor. Looking at the solutions for help should only be done as a last resort, if no one is available to help. In any case, after you get enough help and are able to do the exercise, try redoing it later on your own. And do additional exercises.

If you get a solution to an exercise but it is different from the official solution, take the time to figure out why. Did you make a mistake? Did you forget something? Did you discover another correct solution? If you don’t know, then ask for help from another student or the instructor. If your solution turns out to be incorrect, then try redoing the same exercise later on your own, and do additional exercises.

Feedback on the notes and solutions is welcome. Please send comments to alexis@clarkson.edu.
Chapter 1

Basic Concepts

1.1 A Running Pace Calculator

There are no exercises in this section.

1.2 High-Level Languages and Compilers

There are no exercises in this section.

1.3 Output and Data

1.3.9.

```cpp
#include <iostream>

int main()
{
    std::cout << "Alexis Maciel\n" << "Potsdam, NY\n";

    return 0;
}
```

1
1.4 Structure of a Simple C++ Program

There are no exercises in this section.

1.5 Input and Variables

1.5.7.

```cpp
#include <iostream>
using std::cin;
using std::cout;

int main()
{
    cout << "How old are you? ";

    int age;
    cin >> age;

    cout << "You are " << age << " years old.\n";

    return 0;
}
```
1.6 Arithmetic Operations

1.6.3. (From now on, we will omit the #include directives and the using declarations.)

    int main()
    {
        cout << "What year were you born in? ";
        int year_of_birth;
        cin >> year_of_birth;

        int age = 2012 - year_of_birth;
        cout << "You are " << age << " years old.\n";

        return 0;
    }

1.6.4.

    int main()
    {
        cout << "Temperature in degrees Celcius? ";
        double temp_C;
        cin >> temp_C;

        double temp_F = temp_C * 9 / 5 + 32;
        cout << "The equivalent in degrees Fahrenheit is " << temp_F << ".\n";

        return 0;
    }
1.6.5.

```cpp
int main()
{
    cout << "What is your income? ";
    double income;
    cin >> income;

    cout << "How many dependents do you have? ";
    int num_dependents;
    cin >> num_dependents;

    double tax = (income - 10000 - num_dependents * 5000) * 0.35;
    cout << "Your income tax is " << tax << ".
    << "Your effective tax rate is "
    << tax / income * 100 << "%.

    return 0;
}
```

1.7 More on Variables

There are no exercises in this section.
```
int main() {
    cout << "Temperature in degrees Celcius? ";

    double temp_C;
    cin >> temp_C;

    cout << "Interpretation: ";
    if (temp_C >= 38)
        cout << "very hot";
    else if (temp_C >= 28)
        cout << "hot";
    else if (temp_C >= 18)
        cout << "comfortable";
    else if (temp_C >= -12)
        cout << "cool";
    else
        cout << "very cold";
    cout << ".\n";

    return 0;
}
```
```c++
int main() {
    cout << "Please choose:\n" << " 1. Celcius to Fahrenheit\n" << " 2. Fahrenheit to Celcius\n" << "Choice? (1 or 2) ";

    int choice = 0;
    cin >> choice;
    cout << '\n';

    if (choice == 1) {
        cout << "Temperature in degrees Celcius? ";
        double temp_C;
        cin >> temp_C;
        double temp_F = temp_C * 9 / 5 + 32;
        cout << "The equivalent in degrees Fahrenheit is " << temp_F << "\n";
    } else {
        cout << "Temperature in degrees Fahrenheit? ";
        double temp_F;
        cin >> temp_F;
        double temp_C = (temp_F - 32) * 5 / 9;
        cout << "The equivalent in degrees Celcius is " << temp_C << "\n";
    }

    return 0;
}
```
1.8. CONDITIONAL STATEMENTS

1.8.11.

```c++
int main()
{
    cout << "What is your income? ";

    double income;
    cin >> income;

    cout << "How many dependents do you have? ";

    int num_dependents;
    cin >> num_dependents;

    double taxable_income =
        income - 10000 - num_dependents * 5000;

    double tax = 0;
    if (taxable_income >= 100000)
        tax = 25000 + (taxable_income - 100000) * .40;
    else if (taxable_income >= 50000)
        tax = 10000 + (taxable_income - 50000) * .30;
    else
        tax = taxable_income * .20;

    cout << "Your income tax is " << tax << ".
" << "Your effective tax rate is " << tax / income * 100 << "%.
";

    return 0;
}
```
Chapter 2

Repetition
2.1 Adding a Loop to the Running Pace Calculator

2.1.4.

```cpp
int main() {
    char answer = 'x'; // To question "More?"

    do {
        cout << "Please choose:
        " 1. Celcius to Fahrenheit
        " 2. Fahrenheit to Celcius
        "Choice? (1 or 2) ";

        int choice = 0;
        cin >> choice;
        cout << '
';

        if (choice == 1) {
            ...
        } else {
            ...
        }

        cout << "\nMore? (y or n) ";
        cin >> answer;
        cout << '
';
    } while (answer == 'y');

    cout << "Goodbye!\n";

    return 0;
}
```
2.2. **MORE ON LOOPS**

2.1.5.

```c
int main() {
    int choice = 0;
    do {
        cout << "Please choose:\n"
             << "  1. Celcius to Fahrenheit\n"
             << "  2. Fahrenheit to Celcius\n"
             << "  3. Quit\n"
             << "Choice? ";
        cin >> choice;
        cout << '\n';
        if (choice == 1) {
            ...
        } else if (choice == 2) {
            ...
        } // Else, choice is 3, so do nothing.
        cout << '\n';
    } while (choice != 3);
    cout << "Goodbye!\n";
    return 0;
}
```

2.2 More on Loops

2.2.6. This is probably the easiest solution:

```c
for (int i = 1; i < n; ++i)
    cout << 2*i << ", ";
cout << 2*n << '\n';
```

Here's another solution:
for (int i = 2; i < 2*n; i = i + 2)
    cout << i << ", ";
cout << 2*n << "\n";

2.2.7.

for (int i = 1; i < n; ++i)
    cout << 2*i - 1 << ", ";
cout << 2*n - 1 << "\n";

2.3 Loops and Variables

2.3.2.

int sum = 0;
for (int i = 1; i <= n; ++i) {
    sum += i;
    cout << sum << " " << i*(i+1)/2 << "\n";
}

2.3.3.

int main() {
    cout << "Please enter the various incomes of your household.\n";

double total_income = 0;
int income_nbr = 1;
char answer = 'x'; // To question "Another income?"
do {
    cout << "Income " << income_nbr << " : ";
    double income;
    cin >> income;
    total_income += income;
    ++income_nbr;

    cout << "\nAnother income? (y or n) ";
    cin >> answer;
} while (answer == 'y');
cout << "How many adults in your household? ";
int num_adults;
cin >> num_adults;

cout << "How many dependents in your household? ";
int num_dependents;
cin >> num_dependents;

double taxable_income =
    total_income - num_adults * 10000 - num_dependents * 5000;

double tax = 0;
if (taxable_income >= 100000)
    tax = 25000 + (taxable_income - 100000) * .40;
else if (taxable_income >= 50000)
    tax = 10000 + (taxable_income - 50000) * .30;
else
    tax = taxable_income * .20;

cout << "Your total income is " << total_income << ".
" << "Your taxable income is " << taxable_income << ".
" << "Your income tax is " << tax << ".
" << "Your effective tax rate is " << tax / total_income * 100 << ";";

return 0;
}

2.4 Nested Loops

2.4.3.

for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= i; ++j)
        cout << '*';
    cout << 'n';
}
2.4.4.

```
for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= i - 1; ++j)
        cout << ' ';
    for (int j = i; j <= n; ++j)
        cout << '*';
    cout << '\n';
}
```
Chapter 3

File Input and Output

3.1 A Pay Calculator

There are no exercises in this section.
3.2 File Streams

3.2.7.

```cpp
int main() {
    std::ofstream ofs_hours("hours.txt");

    cout << "Enter employee number 0 when done.\n";

    int employee_number = 0;
    do {
        cout << "\nEmployee number: ";
        cin >> employee_number;

        if (employee_number != 0) {
            double hours = 0;
            cout << "Hours: ";
            cin >> hours;
            ofs_hours << employee_number << ' ' << hours << '\n';
        }
    } while (employee_number != 0);

    return 0;
}
```

3.2.8. The end of the program should be modified as follows:

```cpp
double effective_tax_rate = tax / income * 100;

cout << "\nYour income tax is " << tax << ",\n"
    << "Your effective tax rate is "
    << effective_tax_rate << ".\n";

std::ofstream ofs_return("tax_return.txt");
ofs_return
    << income << '\n'
    << num_dependents << '\n'
    << tax << '\n'
    << effective_tax_rate << '\n';
```
int main() {
    std::ifstream ifs_return("tax_return.txt");

    double income;
    ifs_return >> income;

    int num_dependents;
    ifs_return >> num_dependents;

    double tax;
    ifs_return >> tax;

    double effective_tax_rate;
    ifs_return >> effective_tax_rate;

    cout << "Income: $" << income << '\n'
         << "Number of dependents: " << num_dependents << '\n'
         << "\nIncome tax: $" << tax << '\n'
         << "Effective tax rate: " << effective_tax_rate << "%\n";

    return 0;
}
3.3 Detecting the End of the File

3.3.4.

```c++
int num_employees = 0;
double total_hours = 0;
double total_pay = 0;

int employee_number;
while (ifs_hours >> employee_number) {
    double num_hours;
    ifs_hours >> num_hours;

    double pay = num_hours * 20;
    ofs_pay << employee_number << ' ' << num_hours << ' ' << pay << '\n';

    ++num_employees;
    total_hours += num_hours;
    total_pay += pay;
}

cout << "Average hours: " << total_hours/num_employees << "\nTotal pay: " << total_pay << '\n';
```
3.3. DETECTING THE END OF THE FILE

3.3.5.

```cpp
std::ofstream ofs_return("tax_return.txt");

cout << "Please enter the various incomes of your household.\n";

double total_income = 0;
int income_nbr = 1;
char answer = 'x';  // To question "Another income?"
do {
    cout << "Income " << income_nbr << ": ";
    double income;
    cin >> income;
    total_income += income;
    ++income_nbr;

    ofs_return << income << '\n';

    cout << "\nAnother income? (y or n) ";
    cin >> answer;
} while (answer == 'y');
ofs_return << -1 << '\n';

cout << "\nHow many adults in your household? ";
int num_adults;
cin >> num_adults;

cout << "How many dependents in your household? ";
int num_dependents;
 cin >> num_dependents;

double taxable_income =
    total_income - num_adults * 10000 - num_dependents * 5000;
```
double tax = 0;
if (taxable_income >= 100000)
    tax = 25000 + (taxable_income - 100000) * .40;
else if (taxable_income >= 50000)
    tax = 10000 + (taxable_income - 50000) * .30;
else
    tax = taxable_income * .20;

double effective_tax_rate = tax / total_income * 100;

cout << "\nYour total income is " << total_income << ".\n"
<< "Your taxable income is " << taxable_income << ".\n"
<< "Your income tax is " << tax << ".\n"
<< "Your effective tax rate is "
<< effective_tax_rate << "%.\n";

ofs << num_adults << '
'
<< num_dependents << '
'
<< total_income << '
'
<< taxable_income << '
'
<< tax << '
'
<< effective_tax_rate << '
';
3.3. DETECTING THE END OF THE FILE

3.3.6.

```cpp
std::ifstream ifs_return("tax_return.txt");

int income_nbr = 0;
double income;
ifs_return >> income;
while (income != -1) {
    ++income_nbr;
    cout << "Income " << income_nbr << ": $" << income << 'n';
    ifs_return >> income;
}

int num_adults;
ifs_return >> num_adults;

int num_dependents;
ifs_return >> num_dependents;

double total_income;
ifs_return >> total_income;

double taxable_income;
ifs_return >> taxable_income;

double tax;
ifs_return >> tax;

double effective_tax_rate;
ifs_return >> effective_tax_rate;

cout << "Number of adults: " << num_adults << 
    "Number of dependents: " << num_dependents << 
    "Total income: $" << total_income << 
    "Taxable income: $" << taxable_income << 
    "Income tax: $" << tax << 
    "Effective tax rate: " << effective_tax_rate << 
    ";n";
```
3.3.7.

```cpp
std::ifstream ifs_temperatures("temperatures.txt");

double total_highs = 0;
int num_days = 0;
double high;
while (ifs_temperatures >> high) {
    ++num_days;
    total_highs += high;
}

cout << total_highs/num_days << '\n';
```

3.3.8.

```cpp
std::ifstream ifs_results("results.txt");

int total_yes = 0;
int total_no = 0;

int district_number;
while (ifs_results >> district_number) {
    int num_yes;
    int num_no;
    ifs_results >> num_yes >> num_no;
    total_yes += num_yes;
    total_no += num_no;
}

double total_votes = total_yes + total_no;

cout << "Yes: " << total_yes
    << " (" << total_yes/total_votes*100 << ")\n"
    << "No: " << total_no
    << " (" << total_no/total_votes*100 << ");
```
3.4 Different Wages

3.4.1.

```cpp
std::ifstream ifs_potsdam("potsdam.txt");
std::ifstream ifs_cancun("cancun.txt");

double total_highs_potsdam = 0;
double total_highs_cancun = 0;
int num_days = 0;
double high;
while (ifs_potsdam >> high) {
    ++num_days;
    total_highs_potsdam += high;
    ifs_cancun >> high;
    total_highs_cancun += high;
}

cout << "Potsdam: " << total_highs_potsdam/num_days << 'n'
    << "Cancun: " << total_highs_cancun/num_days << 'n';
```

3.5 String Variables

3.5.3. The beginning of the program should be modified as follows:

```cpp
cout << "Name of temperature file: ";
string temperature_file_name;
getline(cin, temperature_file_name);
std::ifstream ifs_temperatures(temperature_file_name);
```
3.5.4. The beginning of the program is the same. The rest should be changed as follows:

```cpp
double total_highs = 0;
int num_sunny = 0;
int num_mostly_sunny = 0;
int num_mostly_cloudy = 0;
int num_cloudy = 0;
int num_days = 0;
double high;
while (ifs_temperatures >> high) {
    ++num_days;
    total_highs += high;
    ifs_temperatures.get(); // blank space
    string weather;
    getline(ifstd_temperatures, weather);
    if (weather == "sunny")
        ++num_sunny;
    else if (weather == "mostly sunny")
        ++num_mostly_sunny;
    else if (weather == "mostly cloudy")
        ++num_mostly_cloudy;
    else if (weather == "cloudy")
        ++num_cloudy;
}

cout << "High: " << total_highs/num_days << 'n'
    << "Sunny: " << 1.0*num_sunny/num_days * 100 << "%"n
    << "Mostly sunny: "
    << 1.0*num_mostly_sunny/num_days * 100 << "%"n
    << "Mostly cloudy: "
    << 1.0*num_mostly_cloudy/num_days * 100 << "%"n
    << "Cloudy: " << 1.0*num_cloudy/num_days * 100 << "%"n;
```
3.6 Error Checking

3.6.3. The following code should be inserted into the first program right after the declaration of `ofs_return`. (On a Windows computer, you can test this code by creating a file called `tax_return.txt` and changing its properties so that it is read-only.)

```cpp
if (!ofs_return) {
    cout << "\nCould not open file tax_return.txt.\n";
    return 1;
}
```

The following code should be inserted into the second program right after the declaration of `ifs_return`. (You can test this code by deleting the file `tax_return.txt`.)

```cpp
if (!ifs_return) {
    cout << "\nCould not open file tax_return.txt.\n";
    return 1;
}
```

3.7 Extending the Pay Calculator

3.7.1.

```cpp
std::ifstream ifs_temperatures("temperatures.txt");

for (int month = 1; month <= 12; ++month) {
    double total_highs = 0;
    for (int day = 1; day <= 30; ++day) {
        double high;
        ifs_temperatures >> high;
        total_highs += high;
    }
    cout << month << ": " << total_highs/30 << '\n';
}
```
3.7.2.

```cpp
std::ifstream ifs_gradesheet("gradesheet.txt");
std::ofstream ofs_coursegrades("coursegrades.txt");

int num_assessments;
ifs_gradesheet >> num_assessments;

int student_nbr;
while (ifs_gradesheet >> student_nbr) {
    double total_grade = 0;
    for (int i = 1; i <= num_assessments; ++i) {
        int grade;
        ifs_gradesheet >> grade;
        total_grade += grade;
    }
    ofs_coursegrades << student_nbr << ' ' << total_grade/num_assessments << '\n';
}
```
3.7. EXTENDING THE PAY CALCULATOR

3.7.3.

```cpp
std::ifstream ifs_gradesheet("gradesheet.txt");
std::ofstream ofs_coursegrades("coursegrades.txt");

int num_assessments;
ifs_gradesheet >> num_assessments;
ifs_gradesheet.get(); // 'n'

string student_name;
while (getline(ifs_gradesheet, student_name)) {
    double total_grade = 0;
    for (int i = 1; i <= num_assessments; ++i) {
        int grade;
        ifs_gradesheet >> grade;
        total_grade += grade;
    }
    ifs_gradesheet.get(); // 'n'
    ofs_coursegrades << student_name << ' ' << total_grade/num_assessments << 'n';
}
```
3.8 Compile-Time Constants

3.8.5.

```cpp
const string kHoursFileName = "hours.txt";
const string kPayFileName = "pay.txt";
const double kWage = 20;

int main() {
    std::ifstream ifs_hours(kHoursFileName);
    std::ofstream ofs_pay(kPayFileName);

    int employee_number;
    ifs_hours >> employee_number;

    double num_hours;
    ifs_hours >> num_hours;

    double pay = num_hours * kWage;
    ofs_pay << employee_number << ' ' << num_hours << ' ' << pay
    << '
';

    return 0;
}
```

3.9 Formatting of Floating-Point Numbers

3.9.4.

```cpp
cout << fixed << setprecision(0) << "Income: $" << income << '
'
<< "Number of dependents: " << num_dependents << '
'
<< "Income tax: $" << tax << '
'
<< setprecision(1) << "Effective tax rate: " << effective_tax_rate << "\n";
```
Chapter 4

Functions

4.1 Introduction

There are no exercises in this section.

4.2 A Rounding Function

4.2.5.

    double negative(double x)
    {
        return −x;
    }

4.2.6.

    double abs(double x)
    {
        if (x >= 0)
            return x;
        else
            return −x;
    }
CHAPTER 4. FUNCTIONS

4.2.7.

double sum(double x, double y)
{
    return x + y;
}

4.3 Functions in the Pay Calculator

4.3.4.

void print_date(int month, int day, int year, std::ostream & out)
{
    out << month << '/' << day << '/' << year;
}

4.3.5.

void println(std::ostream & out, int x)
{
    out << x << '\n';
}

4.4 Reference Arguments

4.4.9.

void negate(double & x)
{
    x *= -1;
}

4.4.10.

void add(double x, double y, double & z)
{
    z = x + y;
}
4.5. REFERENCE ARGUMENTS IN THE PAY CALCULATOR

4.4.11. $\textbf{void}$ read_date($\text{std}::\text{istream} & \text{in}$, $\text{int} & \text{month}$, $\text{int} & \text{day}$, $\text{int} & \text{year}$)
{
    $\text{char}$ slash;
    in >> month >> slash >> day >> slash >> year;
}

4.4.12. $\textbf{void}$ println($\text{std}::\text{ostream} & \text{out}$, $\text{const}$ $\text{string} & \text{s}$)
{
    $\text{out} << \text{s} << '\n';$
}

4.5 Reference Arguments in the Pay Calculator

There are no exercises in this section.

4.6 Modularity and Abstraction

There are no exercises in this section.

4.7 Documentation

There are no exercises in this section.
Chapter 5

Vectors

5.1 A Simple File Viewer

There are no exercises in this section.

5.2 Vector Basics

5.2.6.

```cpp
v.clear();
for (int i = 1; i <= 100; ++i) v.push_back(10*i);
```

5.2.7.

```cpp
for (const string & s : v) cout << s << '\n';
```

5.2.8.

```cpp
for (int i = v.size() - 10; i < v.size(); ++i)
    cout << v[i] << ' ';
```
5.2.9.

```cpp
int sum(const vector<int> & v)
{
    int total = 0;
    for (int x : v) total += x;
    return total;
}
```

5.2.10.

```cpp
int fill(vector<int> & v, int x)
{
    for (int & y : v) y = x;
}
```

5.2.11.

```cpp
void read(istream & in, vector<int> & v, int n)
{
    v.resize(n);
    for (int i = 0; i < n; ++i) in >> v[i];
}
```

5.2.12.

```cpp
void read(ifstream & ifs, vector<int> & v)
{
    v.clear();
    int x;
    while (ifs >> x) v.push_back(x);
}
```

5.3 Object-Oriented Programming

There are no exercises in this section.
5.4 Design and Implementation of the File Viewer

5.5 Multiway Branches

5.5.2.

```cpp
int main() {
    int choice = 0;
    do {
        cout << "Please choose:
        " 1. Celcius to Fahrenheit
        " 2. Fahrenheit to Celcius
        " 3. Quit
        "Choice? ";
        cin >> choice;
        cout << '
';

        switch (choice) {
            case 1:
                ...
                break;
            case 2:
                ...
                break;
        }
        cout << '
';
    } while (choice != 3);
    cout << "Goodbye! 
";
    return 0;
}
```
CHAPTER 5. VECTORS

5.6 More on Vectors

5.6.3.

```cpp
vector<int> v0;
cout << "An empty vector of integers: ";
for (int i = 0; i < v0.size(); ++i) cout << v0[i] << ' ';
cout << '\n';

vector<int> v1(3);
cout << "A vector with three (random) integers: ";
for (int i = 0; i < v1.size(); ++i) cout << v1[i] << ' ';
cout << '\n';

vector<string> v2(3);
cout << "A vector with three empty strings: ";
for (int i = 0; i < v2.size(); ++i)
    cout << "'" << v2[i] << "' ";
cout << '\n';

vector<int> v3(3, 17);
cout << "A vector with three 17's: ";
for (int i = 0; i < v3.size(); ++i) cout << v3[i] << ' ';
cout << '\n';

vector<int> v4(v3);
cout << "A copy of the previous vector: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '\n';

v4.front() = 1;
v4.back() = 23;
cout << "The last vector with its first and last elements changed to 1 and 23: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '\n';
```
v4.resize(2);
cout << "The last vector shrunk to size 2: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '
';

v4.resize(4);
cout << "The last vector grown to size 4: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '
';

v4.resize(6, 42);
cout << "The last vector grown to size 6 and padded with "
   << "42’s: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '
';

v4.push_back(60);
cout << "The last vector with a 60 added to its back: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '
';

v4.pop_back();
cout << "The last vector with its back element removed: ";
for (int i = 0; i < v4.size(); ++i) cout << v4[i] << ' ';
cout << '
';

vector<int> v5;
vector<int> v6;
v5 = v6 = v4;
cout << "Two new copies of the last vector: 
";
for (int i = 0; i < v5.size(); ++i) cout << v5[i] << ' ';
cout << '
';
for (int i = 0; i < v6.size(); ++i) cout << v6[i] << ' ';
cout << '
';

v6.clear();
cout << "The last vector with all its elements removed: ";
for (int i = 0; i < v6.size(); ++i) cout << v6[i] << ' ';
cout << '
';
cout << "The last vector is empty: ";
if (v6.empty())
    cout << "true";
else
    cout << "false";
cout << ' \n';

cout << "The other one before that is empty: ";
if (v5.empty())
    cout << "true";
else
    cout << "false";
cout << ' \n';

cout << "The maximum size of the last vector: ";
    << v6.max_size() << ' \n';

v6.assign(5, 12);
cout << "The last vector with five 12’s: ";
for (int i = 0; i < v6.size(); ++i) cout << v6[i] << ' ';
cout << ' \n';

cout << "The last two vectors:\n";
for (int i = 0; i < v5.size(); ++i) cout << v5[i] << ' ';
cout << ' \n';
for (int i = 0; i < v6.size(); ++i) cout << v6[i] << ' ';
v5.swap(v6);
cout << "The same vectors with their contents swapped:\n";
for (int i = 0; i < v5.size(); ++i) cout << v5[i] << ' ';
cout << ' \n';
for (int i = 0; i < v6.size(); ++i) cout << v6[i] << ' ';
cout << ' \n';
5.7  More on Strings

5.7.4.

```cpp
void println(const string & s) {
    for (int i = 0; i < s.length(); ++i)
        cout << s[i];
    cout << '\n';
}
```

5.7.5.

```cpp
string s = "John Doe";
string s2;
int ix_space = s.find(' ');

// Copy last name
for (int i = ix_space + 1; i < s.length(); ++i)
    s2 += s[i];

s2 += "", ";

// Copy first name
for (int i = 0; i < ix_space; ++i)
    s2 += s[i];
```

5.8  A Simple Text Editor

5.9  Adding More Error-Checking to the File Viewer

5.10  Arrays

5.10.5.

```cpp
for (int i = 1; i <= N; ++i) a[i-1] = i*10
```
5.10.6.

```cpp
int sum(const int a[], int n)
{
    int total = 0;
    for (int i = 0; i < n; ++i) total += a[i];
    return total;
}
```

5.10.7.

```cpp
int fill(int a[], int n, int x)
{
    for (int i = 0; i < n; ++i) a[i] = x;
}
```

5.10.8.

```cpp
void read(istream & in, int a[], int n)
{
    for (int i = 0; i < n; ++i) in >> a[i];
}
Chapter 6

Structures

6.1 Extending the Pay Calculator

6.1.3.

```c
void init(Time & t, int hours, int minutes)
{
    t.hours = hours;
    t.minutes = minutes;
}
```

6.1.4.

```c
struct Date
{
    int month = 1;
    int day = 1;
    int year = 2000;
};

void init(Date & d, int month, int day, int year)
{
    d.month = month;
    d.day = day;
    d.year = year;
}
```
std::istream & read(Date & d, std::istream & in)
{
    in >> d.month;
    in.get(); // '/'
    in >> d.day;
    in.get(); // '/'
    in >> d.year;
    in.get(); // '/'
    return in;
}

void print(const Date & d, std::ostream & out)
{
    out << d.month << '/' << d.day << '/' << d.year;
}

void print_in_words(const Date & d, std::ostream & out)
{
    const string kaMonthNames[12] = {
    out << kaMonthNames[d.month - 1] << ' ' << d.day << ',' << d.year;
}

6.1.5.

struct ThreeDVector
{ double x = 0;
  double y = 0;
  double z = 0;
};
6.1. EXTENDING THE PAY CALCULATOR

```cpp
void init(ThreeDVector & v,
          double x0, double y0, double z0)
{
    v.x = x0;
    v.y = y0;
    v.z = z0;
}

std::istream & read(ThreeDVector & v, std::istream & in)
{
    char dummy;
    in >> dummy; // '('.
    // >> used instead of get() so whitespace is skipped.
    in >> v.x;
    in >> dummy; // ','
    in >> v.y;
    in >> dummy; // ','
    in >> v.z;
    in >> dummy; // ')
    return in;
}

void print(const ThreeDVector & d, std::ostream & out)
{
    out << '(' << d.x << ', ' << d.y << ', ' << d.z << ')';
}

ThreeDVector add(const ThreeDVector & v1,
                  const ThreeDVector & v2)
{
    ThreeDVector sum;
    init(sum, v1.x + v2.x, v1.y + v2.y, v1.z + v2.z);
    return sum;
}
```
6.1.6.

```c
struct Fraction
{
    int a = 0;
    int b = 1;
};

void initialize(Fraction & r, int a0)
{
    r.a = a0;
    r.b = 1;
}

void initialize(Fraction & r, int a0, int b0)
{
    r.a = a0;
    r.b = b0;
}

void read(Fraction & r, istream & in)
{
    in >> r.a;
    in.get(); // '/'
    in >> r.b;
}

void print(const Fraction & r, ostream & out)
{
    out << r.a << '/' << r.b;
}
```
6.2. **Improving the Design of the Text Editor**

```cpp
void print_mixed(const Fraction & r, ostream & out) {
    int n;
    int a = r.a;
    int b = r.b;

    if (b < 0) { a = -a; b = -b; }

    n = a / b;
    a = a % b;

    out << n;
    if (a > 0)
        out << ' ' << a << '/' << b;
    else if (a < 0)
        out << ' ' << -a << '/' << b;
    // else (a == 0), don’t print fraction
}

Fraction add(const Fraction & r1, const Fraction & r2) {
    Fraction result;
    initialize(result, r1.a * r2.b + r2.a * r1.b, r1.b * r2.b);
    return result;
}

Fraction multiply(const Fraction & r1, const Fraction & r2) {
    Fraction result;
    initialize(result, r1.a * r2.a, r1.b * r2.b);
    return result;
}
```

### 6.2 Improving the Design of the Text Editor


Chapter 7

Algorithms and Generic Programming

7.1 Introduction

There are no exercises in this section.

7.2 Generic Programming

7.2.5.

// Requirement on T: values can be compared by using the // less-than operator (<).

```cpp
template <typename T>
T min(T x, T y)
{
    if (x < y)
        return x;
    else
        return y;
}
```
CHAPTER 7. ALGORITHMS AND GENERIC PROGRAMMING

// Requirement on T: values can be added by using the // addition operator (+).
template <typename T>
T average(T x, T y)
{
    return (x + y)/2;
}

7.3 Some Simple Algorithms

7.3.1. Because elements in the subrange [dest,stop) would be overwritten before they are copied. For example, if a vector contains 1 2 3 4 5 6 and we tried to copy the elements 1 2 3 4 to the position where 3 is, then the result would be 1 2 1 2 1 2, not 1 2 1 2 3 4.

7.3.2.

// Requirement on T: values can be printed to cout by using the // output operator (<<).
template <typename T>
void println(const vector<T> & v, int start, int stop)
{
    for (int i = start; i < stop; ++i)
        cout << v[i] << ''
        cout << '\n';
}

// Requirement on T: values can be added and 0 is a value of // type T.
template <typename T>
T sum(const vector<T> & v, int start, int stop)
{
    T total = 0;
    for (int i = start; i < stop; ++i)
        total += v[i];
    return total;
}
7.3. SOME SIMPLE ALGORITHMS

// Requirement on T: values can be added, values can be divided
// by an integer and 0 is a value of type T.
template <typename T>
T average(const vector<T> & v, int start, int stop)
{
    T total = 0;
    for (int i = start; i < stop; ++i)
        total += v[i];
    return total / (stop - start);
}

template <typename T>
void fill(vector<T> & v, int start, int stop, const T & e)
{
    for (int i = start; i < stop; ++i) v[i] = e;
}

// Requirement on T: values can be compared by using the ==
// operator.
template <typename T>
void replace(vector<T> & v, int start, int stop, const T & x, const T & y)
{
    for (int i = start; i < stop; ++i)
        if (v[i] == x) v[i] = y;
}

// Precondition: assumes that the range is not empty.
// Requirement on T: values can be compared by using the <
// operator.
template <typename T>
int min_element(const vector<T> & v, int start, int stop)
{
    int ix_min = start;
    for (int i = start + 1; i < stop; ++i)
        if (v[i] < v[ix_min]) ix_min = i;
    return ix_min;
}
template <typename T>
void reverse(vector<T> & v, int start, int stop)
{
    int i = start;
    int j = stop - 1;
    while (i < j)
    {
        std::swap(v[i], v[j]);
        ++i;
        --j;
    }
}

template <typename T>
int copy_backward(const vector<T> & v1, int start, int stop,
                 vector<T> & v2, int dest)
{
    for (int i = stop - 1; i >= start; --i) {
        --dest;
        v2[dest] = v1[i];
    }
    return dest;
}

7.4 Algorithms in the STL