Learning C++ **Summer 2025** **Mobina Amrollahi**

Exercise 1

1. Briefly describe the design aims of C++ and comment on the extent to which C++ meets those.
   Answer. C++ was designed to combine the strengths of C as a systems programming language with Simula's facilities for organizing programs. To meet these two objectives, C++ was designed to be a *general-purpose programming language* with a bias towards *systems programming* that

   - is a better C
   - supports data abstraction
   - supports object-oriented programming
   - supports generic programming

2. What are the main programming styles supported by C++?
   Answer. C++ supports:

   - data abstraction, i.e., classes
   - object-oriented programming, i.e., inheritance and polymorphism
   - generic programming, i.e., templates

3. Check the following page: www.stroustrup.com/applications and list 20 major real-world C++ applications.
   Answer.

   (a) **Adobe Systems**: All major applications are developed in C++:
   - Photoshop & ImageReady
   - Illustrator
   - Acrobat
   - InDesign
   - GoLive
   - Frame (mostly C, some C++)

   (b) **Amazon.com**: Software for large-scale e-commerce.

   (c) **Apple**: OS X is written in a mix of languages, but a few important parts are in C++:
   - Finder
   - IOKit device drivers (IOKit is the only place where C++ is used in the kernel.)

   (d) **AT&T**: The largest US telecommunications provider.
   - 1-800 service
   - Provisioning systems
   - Systems for rapid network recovery after failure

   (e) **Autodesk**: A large number of major applications in the CAD domain.

   (f) **Bloomberg**: Provides real-time financial information to investors.

(g) **CERN**: Data analysis for large high-energy physics experiments using the ROOT toolset and libraries.

(h) **Ericsson**:

- TelORB – distributed operating system with object-oriented distributed RAM database
- Base for the TSP application server platform
- TDMA-CDMA HLR
- GSM-TDMA-CDMA mobility gateway
- AAA server

(i) **Facebook (now Meta)**: Several high-performance and high-reliability components.

(j) **Google**:

- Web search engine
- Chromium browser
- Google File System
- MapReduce large-cluster data processing

(k) **HP (Hewlett-Packard)**: A small sample of HP's C++ applications:

- C, C++, Fortran90 compilers, and linker for the HP IA64 platform (over 1M lines of C++)
- SAM (system management utility)
- Networking libraries in HP-UX
- Java VM core
- Parts of OpenView
- Non-stop XML parser (originally from Compaq)

(l) **IBM**:

- OS/400
- K42 – high-performance, open-source OS kernel for cache-coherent multiprocessors

(m) **Intel**:

- VTune performance analysis software
- Compilers and optimizers
- Various chip design and manufacturing software

(n) **Microsoft**: Most products are built using Visual C++. Major products include:

- Windows XP, Vista, 7, NT (NT4, 2000), 95, 98, Me
- Microsoft Office (Word, Excel, Access, PowerPoint, Outlook)
- Internet Explorer and Outlook Express
- Visual Studio (C++, VB, FoxPro)
- Exchange, SQL
- Minor products: FrontPage, Money, Picture It, Project
- Z3 (open-source theorem prover from Microsoft Research)
- All Microsoft games

(o) **MongoDB**: An open-source database used widely in web applications and large enterprises (e.g., Viacom, Disney).

(p) **Morgan Stanley**: C++ is used extensively for financial modeling.

(q) **Mozilla**: Firefox browser and Thunderbird mail client.

(r) **MySQL**: MySQL Server (about 250,000 lines of C++) and MySQL Cluster.

(s) **NASA**:

- Mars rover autonomous driving system (scene analysis and route planning)
- James Webb Telescope software
- International Space Station software components

(t) **Games**: *Warcraft III*, *World of Warcraft*.

Exercise 2

1. What does a compiler do? What does a linker do?
   Answer. A compiler processes the source text of C++ program, and further returns object files which are combined through a linker resulting in an executable program.

2. Change the *Hello, world!* program to output the two lines:

   ```
   Hello, programming!
   Here you go!
   ```

   Answer.

   ```
   #include <iostream>
   int main() {

              std::cout << "Hello, programming! \n";
              std::cout << "Here you go! \n";

              return 0;
              }
   ```

3. Is this a valid program? Why or why not?

   ```
   #include <iostream>
   int main() { std::cout << "Hello, world!" << std::endl; }
   ```

   Answer. Yes, this is a valid program. Since no value is returned, (also known as Void) the system interprets this program as a successful one with the main function returning nothing.

4. Is this a valid program? Why or why not?

   ```
   #include <iostream>
   int main() std::cout << "Hello, world!" << std::endl;
   ```

   Answer. No, this is not a valid program. The systems expects curly braces after any function defined, such as the "main" in this situation.

5. What is the shortest valid program?
   Answer.

```
#include <iostream>
int main() {}
```

   There's no return value from the main function to the system which is interpreted as success.

Exercise 3

1. Is this a valid program? Why or why not?

```
#include <iostream>
#include <string>
int main()
{
    /* This is a comment that extends over several lines
       because it uses /* and */ as its starting and
       ending delimiters */
    std::cout << "Does this work?\n";
    return 0;
}
```

   Answer. This is not a valid C++ program because every multi-line comment starts with /* and ends at the first occurrence of */. In this case, the */ appearing after the word "and" is treated by the compiler as the end of the comment. As a result, the remainder of the sentence is interpreted as actual code, which leads to a compilation error since it contains invalid syntax.

2. Is this a valid program? Why or why not?

```
#include <iostream>
#include <string>
int main()
{
    // This is a comment that extends over several lines
    // by using // at the beginning of each line instead of using /*
    // or */ to delimit comments.
    std::cout << "Does this work?\n";
    return 0;
}
```

   Answer. This is the valid program. The // at the beginning of the first three lines in the main function indicates comments, causing the compiler to ignore those lines. As a result, only the std::cout statement is executed.

3. Write a program that, when run, writes the Hello, world!" program as its output.
Answer.

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

4. Read two integer numbers and compute their average. Re-write the program for 3, 4, and 5 numbers. Do you think we can extend programs this way? What is your conclusion?
Answer. Extending the program in this way isn't efficient, especially if the growth rate is faster than linear. This highlights the usefulness of loops: with each additional integer, both the number of inputs and the formula for calculating the average update automatically—eliminating the need for the programmer to adjust the code manually each time.

(a) **Read two integer numbers and compute their average**

```
#include <iostream>
#include <string>
int main()
{
    int num1, num2;
    std::cin num1;
    std::cin num2;
    std::cout << "The average is:" << (num1 + num2) / 2 << '\n';
    return 0;
}
```

(b) **Read three integer numbers and compute their average**

```
#include <iostream>
#include <string>
int main()
{
    int num1, num2, num3;
    std::cin num1;
    std::cin num2;
    std::cin num3;
    std::cout << "The average is:" << (num1 + num2 + num3) / 3 << '\n';
    return 0;
}
```

(c) **Read four integer numbers and compute their average**

```
#include <iostream>
#include <string>
int main()
{
    int num1, num2, num3, num4;
    std::cin num1;
```

```
    std::cin num2;
    std::cin num3;
    std::cin num4;
    std::cout << "The average is:" << (num1 + num2 + num3 + num4) / 4 << '\n
        ';
    return 0;
}
```

(d) **Read five integer numbers and compute their average**

```
#include <iostream>
#include <string>
int main()
{
    int num1, num2, num3, num4, num5;
    std::cin num1;
    std::cin num2;
    std::cin num3;
    std::cin num4;
    std::cin num5;
    std::cout << "The average is:" << (num1 + num2 + num3 + num4 + num5) / 5
         << '\n';
    return 0;
}
```

Exercise 4

1. Why are the compound assignment operators in the form:

   `expr1 op= expr2`

   rather than

   `expr1 =op expr2`?

   For example, we write

   `x -= y`

   rather than

   `x =- y`

   Answer. The second form is ambiguous, as it's unclear whether we intend to assign the negative of y to x, or subtract y from x and then assign the result back to x.

Exercise 5

1. **Distance between two points:** Write a program that reads the coordinates of two points, then computes the distance between them. Use a proper prompt for reading two points. Use the following formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Note: Use the `sqrt` function from the `<cmath>` header.

Answer.

```
// distance_between_points.cpp
#include <iostream>
#include <cmath>

int main()
{
    double x1, y1;
    std::cout << "Enter the coordinates of first point: \n";
    std::cin >> x1 >> y1;

    double x2, y2;
    std::cout << "Enter the coordinates of the second point: \n";
    std::cin >> x2 >> y2;

    double dist = sqrt(pow((x1 - x2), 2) + pow((y1 - y2), 2));
    std::cout << "The distance is " << dist << '\n';
    return 0;
}
```

2. **Simple calculator:** Write a program that reads two lines of input. At the first line an arithmetic operation (`+`, `-`, `*`, `/`, `%`). At the second line, you should offer two integers and do the desired arithmetic operation.

   Example:

   ```
   operator (+, -, *, /, %): *
   operands: 3 6
   result: 18
   ```

   Answer.

   ```
   // calculator.cpp
   #include <iostream>
   #include <set>
   #include <string>

   int main()
   {
       std::set<char> viable_operations = {'+', '-', '/', '*', '%'};

       std::string oper; // The user might provide other than a charcter
       std::cout << "Enter the arithmetic operation: \n";
       std::cin >> oper;

       if (oper.length() != 1 || viable_operations.find(oper[0]) == viable_operations.
           end()) {
   ```

```
        std::cout << "The arithmetic operation is not valid. \n";
        return 1;
    }

    char op = oper[0];
    long x, y;
    std::cout << "Enter two numbers: \n";
    std::cin >> x >> y;

    long result;
    if (op == '+') result = x + y;
    else if (op == '-') result = x - y;
    else if (op == '*') result = x * y;
    else if (op == '/') {
        if (y == 0) {
            std::cout << "Division by zero! \n";
            return 1;
        }
        result = x / y;
    }
    else if (op == '%') result = x % y;

    std::cout << x << ' ' << op << ' ' << y << " = " << result << '\n';
    return 0;
}
```

Exercise 6

1. Write a program to count down from 10 to -5? **Answer.**

```
    // count down from 10 to -5
    #include <iostream>

    int main()
    {
        int i = 10;
        while (i > -6){
            std::cout << i << '\n';
            --i;
        }
        return 0;
    }
```

2. What does the following code do?

```
    int i = 0;
    while (i < 10) {
        i += 1;
        std::cout << i << '\n';
    }
```

Answer. This program will print numbers from 1 to 10 on separate lines.

3. Write a program to generate the product of the numbers in the range of $[1, 10)$. **Answer.**

```
// calculate product of numbers in range [1,10)
#include <iostream>

int main()
{
    int i = 1;
    int product = 1;
    while (i < 10){
        product = product * i;
        i += 1;
    }
    std::cout << "Product is: " << product << '\n';
    return 0;
}
```

Exercise 7

1. What does the following example do?

```
// Duff's device. Helpful comment deliberately deleted.
void send(int* to, int* from, int count)
{
    int n = (count+7)/8;
    switch (count % 8) {
    case 0: do { *to++ = *from++;
    case 7: *to++ = *from++;
    case 6: *to++ = *from++;
    case 5: *to++ = *from++;
    case 4: *to++ = *from++;
    case 3: *to++ = *from++;
    case 2: *to++ = *from++;
    case 1: *to++ = *from++;
           } while (--n > 0);
    }
}
```

Why would anyone write something like that? No, this is not recommended as good style.
Answer

```
#include <iostream>
#include <cstdlib>

void send(int* to, int* from, int count);

int main()
{
    int size = 20;
    int dest[size], src[size];

    for (int i = 0; i < size; ++i)
        src[i] = std::rand() % 2;
```

```
    send(dest, src, size);

    for (int i = 0; i < size; ++i)
        std::cout << src[i] << '\t' << dest[i] << '\n';
    return 0;
}

void send(int* to, int* from, int count)
{
    int n = (count + 7) / 8;
    switch (count % 8) {
        case 0: do { *to++ = *from++;
        case 7: *to++ = *from++;
        case 6: *to++ = *from++;
        case 5: *to++ = *from++;
        case 4: *to++ = *from++;
        case 3: *to++ = *from++;
        case 2: *to++ = *from++;
        case 1: *to++ = *from++;
                } while (--n > 0);
    }
}
```

Duff's device is a way of copying elements from one array to another. It uses a switch + do-while loop to "unroll" the copy operation, so fewer loop iterations are needed. Back in the 1980s, C compilers didn't optimize loops well. This trick reduced loop overhead by manually unrolling eight copies per iteration. Now that compilers are more efficient in unrolling a loop, we can simply iterate through the source array and copy it to the destination. Applications of suc unrollment are printers, and how digital text can be copied to physical paper.

Exercise 8

1. Write a function called `power` that takes two integers called `base` and `exp` and computes the value of `base` raised to the power `exp` and **returns** it. Call `power` with various parameters. Try to handle large numbers. **Hint:** Use `long long` as the return type.
   Answer.

```
// Example program for power function
#include <iostream>
#include <cmath>

long long power(int /* base */, int /* exp */);

int main()
{
    int base, exp;
    std::cout << "Enter the base: \n";
    std::cin >> base;
    std::cout << "Enter the exponent: \n";
    std::cin >> exp;
```

```
    long long result = power(base, exp);
    std::cout << "The result is: " << result << '\n';
    return 0;
}

long long power(int base, int exp)
{
    return pow(base, exp);
}
```

2. Write a program that prompts the user to enter three integer values, and then outputs the values in numerical sequence separated by commas. So, if the user enters the values 10 4 6, the output should be 4, 6, 10. If two values are the same, they should just be ordered together. So, the input 4 5 4 should give 4, 4, 5. **Note:** Don't use arrays and sort algorithms.
Answer

```
#include <iostream>
void sort_three(int /* 1st int */, int /* 2nd int */, int /* 3rd int*/);

int main()
{
    int first, second, third;
    std::cout << "Enter the three numbers: \n";
    std::cin >> first >> second >> third;
    sort_three(first, second, third);
    return 0;
}

void sort_three(int first, int second, int third)
{
    if (first <= second)
    {
        if (first <= third)
        {
            if (second <= third) std::cout << first << "," << second << "," <<
                third << '\n';
            else std::cout << first << "," << third << "," << second << '\n';
        }
        else std::cout << third << "," << first << "," << second << '\n';
    }
    else
    {
        if (first <= third) std::cout << second << "," << first << "," << third
            << '\n';
        else
        {
            if (second <= third) std::cout << second << "," << third << "," <<
                first << '\n';
            else std::cout << third << "," << second << "," << first << '\n';
        }
    }
}
```

Exercise 9

1. Is the following definition valid? Why or why not?

```
const std::string hello = "Hello";
const std::string message = hello + ", world" + "!";
```

Answer.
Yes. Adding one constant string to another is valid. If we print the `message` string, the output will be "Hello, world!".