

Astroinformatics II (Semester 2 2024)

Introduction to C/C++ (I)

Nina Hernitschek

Centro de Astronomía CITEVA
Universidad de Antofagasta

October 15, 2024

Motivation

Python is a versatile modern programming language commonly used. Usually, it is also executing reasonably fast.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Motivation

Python is a versatile modern programming language commonly used. Usually, it is also executing reasonably fast.



But: Sometimes fractions of a second count when carrying out operations on huge amounts of data.

Using the programming language C/ C++ can be $\sim 10^2 - 10^3 \times$ faster on certain operations.

What is a millisecond on one data set, might **save you months of computing time** on data sets like **complete all-sky surveys**.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Motivation

Python is a versatile modern programming language commonly used. Usually, it is also executing reasonably fast.



But: Sometimes fractions of a second count when carrying out operations on huge amounts of data.

Using the programming language C/ C++ can be $\sim 10^2 - 10^3 \times$ faster on certain operations.

What is a millisecond on one data set, might **save you months of computing time** on data sets like **complete all-sky surveys**.

(My largest speed-up: factor of $\sim 10^2 - 10^3$ by replacing an integral in Python with C code. This changed the situation from "impossible" to "runs in 1 month on a medium-sized cluster".)

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

What is C++?

C++ is a **general-purpose programming language** that was developed as an enhancement of the C language to include **object-oriented** paradigm. It is a **compiled** language.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

What is C++?

C++ is a **general-purpose programming language** that was developed as an enhancement of the C language to include **object-oriented** paradigm. It is a **compiled** language.

C++ is a cross-platform language, giving programmers a high level of control over **system resources and memory**.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

What is C++?

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ is a **general-purpose programming language** that was developed as an enhancement of the C language to include **object-oriented** paradigm. It is a **compiled** language.

C++ is a cross-platform language, giving programmers a high level of control over **system resources and memory**.

C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.

What is C++?

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ is a **general-purpose programming language** that was developed as an enhancement of the C language to include **object-oriented** paradigm. It is a **compiled** language.

C++ is a cross-platform language, giving programmers a high level of control over **system resources and memory**.

C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.

Difference between C and C++:

C++ was developed as an extension of C, and both languages have almost the same syntax.

The main difference between C and C++ is that C++ support classes and objects, while C does not.

We use here generally C++, and introduce object-orientated concepts later on.

Why Learn and Use C++?

C++ is portable and can be used to develop applications that can be adapted to multiple platforms.

As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

With C++ supporting low-level, system-level programming, it is suitable for developing operating systems, device drivers, and other system software operating close to hardware.

C++ also provides a rich set of libraries and features for high-level application programming, making it a popular choice for developing scientific software, desktop applications, video games, and other complex applications.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Why Learn and Use C++?

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ is portable and can be used to develop applications that can be adapted to multiple platforms.

As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

With C++ supporting low-level, system-level programming, it is suitable for developing operating systems, device drivers, and other system software operating close to hardware.

C++ also provides a rich set of libraries and features for high-level application programming, making it a popular choice for developing scientific software, desktop applications, video games, and other complex applications.



Overall, C++ is a powerful and versatile programming language that is widely used for a range of applications.

A simple Code Example

Here is a simple C++ code **example** to help you understand the language:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout << "Hello, World!" << endl;
6      return 0;
7  }
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

A simple Code Example

Here is a simple C++ code **example** to help you understand the language:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout << "Hello, World!" << endl;
6      return 0;
7  }
```

Output:

```
Hello, World!
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

A simple Code Example

Here is a simple C++ code **example** to help you understand the language:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout << "Hello, World!" << endl;
6      return 0;
7  }
```

Output:

```
Hello, World!
```

Example explained:

Line 1: `#include <iostream>` is a header file library for input and output objects, such as `cout` (used in line 5). Header files add functionality.

Line 2: `using namespace std` means that we can use names for objects and variables from the standard library.

Line 3: A blank line. C++ ignores white space. But we use it to make the code more readable.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

A simple Code Example

Here is a simple C++ code **example** to help you understand the language:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout << "Hello, World!" << endl;
6      return 0;
7  }
```

Output:

```
Hello, World!
```

Example explained:

Line 4 to 7: The main function `int main()` is the start point into the program. Function code is enclosed in curly brackets `{}`.

Line 5: `cout` (pronounced "c-out") is an object used together with the insertion operator `<<` to output/print text. In our example, it will output `Hello World!`. **Note:** Every C++ statement ends with a semicolon `;`.

Line 6: `return 0;` ends the main function.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Getting Started

To start using C++, you need two things:

- A text editor, like Notepad, to write C++ code.
- A compiler, like GCC, to translate the C++ code into a language that the computer will understand.

Remember: C++ is a compiled language, not a interpreted language.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Getting Started

To start using C++, you need two things:

- A text editor, like Notepad, to write C++ code.
- A compiler, like GCC, to translate the C++ code into a language that the computer will understand.

Remember: C++ is a compiled language, not a interpreted language.

Optional: IDE (such as VSCode)

An **integrated development environment (IDE)** is a software application that helps programmers develop software code efficiently by combining capabilities such as software editing, compiling, debugging, testing in one easy-to-use application.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

A computer program is a list of instructions to be executed by a computer. In a programming language, these programming instructions are called **statements**.

example: The following statement instructs the compiler to print the text "Hello World" to the screen.

```
cout << "Hello World!";
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

A computer program is a list of instructions to be executed by a computer. In a programming language, these programming instructions are called **statements**.

example: The following statement instructs the compiler to print the text "Hello World" to the screen.

```
cout << "Hello World!";
```

It is important that you end the statement with a semicolon ;
If you forget the semicolon, an error will occur and the program will not be compiled.

example:

```
cout << "Hello World!"  
error: expected ';' before 'return'
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

The `cout` object, together with the `<<` operator, is used to output values/print text.

example:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!";
    return 0;
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

You can add as many `cout` objects as you want. However, note that it does not insert a new line at the end of the output.

example:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!";
    cout << "I am learning C++";
    return 0;
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

You can insert a new line with the `endl` manipulator:

example:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!" << endl;
    cout << "I am learning C++";
    return 0;
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

Another way to insert a new line into the output is using the `\n` character:

example:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World! \n";
    cout << "I am learning C++";
    return 0;
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

Another way to insert a new line into the output is using the `\n` character:

example:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World! \n";
    cout << "I am learning C++";
    return 0;
}
```

Two `\n` will produce an **empty line**.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

Another way to insert a new line into the output is using the `\n` character:

example:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World! \n";
    cout << "I am learning C++";
    return 0;
}
```

Two `\n` will produce an **empty line**.

What is `\n` exactly?

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

The **newline character** `\n` is called an **escape sequence**. It forces the cursor to go to the beginning of the next line on the screen.

The full list of escape sequences is given in the following:

| Escape Sequence | Description |
|-----------------|--|
| <code>\n</code> | new line |
| <code>\t</code> | horizontal tab |
| <code>\v</code> | vertical tab |
| <code>\f</code> | form feed |
| <code>\r</code> | carriage return |
| <code>\\</code> | backslash character (<code>\</code>) |
| <code>\"</code> | double quote character |
| <code>\'</code> | single quote character |
| <code>\?</code> | question mark character |
| <code>\b</code> | backspace |
| <code>\a</code> | audible bell (beeps) |

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Statements

The **newline character** `\n` is called an **escape sequence**. It forces the cursor to go to the beginning of the next line on the screen.

The full list of escape sequences is given in the following:

| Escape Sequence | Description |
|-----------------|--|
| <code>\n</code> | new line |
| <code>\t</code> | horizontal tab |
| <code>\v</code> | vertical tab |
| <code>\f</code> | form feed |
| <code>\r</code> | carriage return |
| <code>\\</code> | backslash character (<code>\</code>) |
| <code>\"</code> | double quote character |
| <code>\'</code> | single quote character |
| <code>\?</code> | question mark character |
| <code>\b</code> | backspace |
| <code>\a</code> | audible bell (beeps) |

It is obvious that these escape sequences were designed for the early typewriter-like text terminals.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Comments

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Comments can be used to explain C++ code, and to make it more readable. It can also be used to prevent execution when testing alternative code. Comments can be single-lined or multi-lined.

Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by the compiler (will not be executed).

This **example** uses a single-line comment before a line of code:

```
// This is a comment  
cout << "Hello World!";
```

This **example** uses a single-line comment at the end of a line of code:

```
cout << "Hello World!"; // This is a comment
```

C++ Comments

Multi-line comments start with `/*` and end with `*/`.

example:

```
/* The code below will print the words Hello World!  
to the screen for testing purposes. */  
cout << "Hello World!";
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

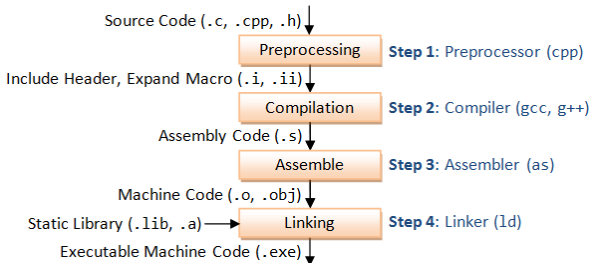
Variables

C++
Operators

Summary &
Outlook

Compiling C/C++ Programs

Other than Python, C/C++ is a **compiled** programming language. C/C++ compilers use a multi-step compilation process to generate executable files from source code files:



Preprocessing: Preprocess the source program (such as a .c file) to generate an .i file.

Compilation: Compile the preprocessed .i file into an assembly language to generate an .s file.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

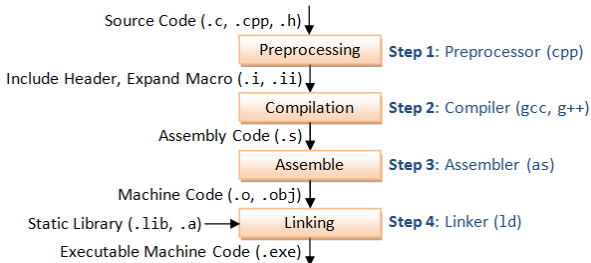
Variables

C++
Operators

Summary &
Outlook

Compiling C/C++ Programs

Other than Python, C/C++ is a **compiled** programming language. C/C++ compilers use a multi-step compilation process to generate executable files from source code files:



Assembling: Assemble the assembly language file to generate the target file .o.

Linking: Link the .o files of each module to generate an executable program file.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

There are many compilers for C/C++ available. We're using here the **GCC compiler**.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

There are many compilers for C/C++ available. We're using here the **GCC compiler**.

The GNU Compiler Collection (GCC) is a powerful and high-performance multi-platform compiler developed by GNU. The GCC compiler can compile and link source programs, assemblers, and target programs of C and C++ into executable files. By default, the GCC software package is installed with Linux.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

There are many compilers for C/C++ available. We're using here the **GCC compiler**.

The GNU Compiler Collection (GCC) is a powerful and high-performance multi-platform compiler developed by GNU. The GCC compiler can compile and link source programs, assemblers, and target programs of C and C++ into executable files. By default, the GCC software package is installed with Linux.

GCC is portable and run in many operating platforms. GCC (and GNU Toolchain) is currently available on all Unixes. They are also ported to Windows (by Cygwin, MinGW and MinGW-W64). GCC is also a cross-compiler, for producing executables on different platform.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

There are many compilers for C/C++ available. We're using here the **GCC compiler**.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

The GNU Compiler Collection (GCC) is a powerful and high-performance multi-platform compiler developed by GNU. The GCC compiler can compile and link source programs, assemblers, and target programs of C and C++ into executable files. By default, the GCC software package is installed with Linux.

GCC is portable and run in many operating platforms. GCC (and GNU Toolchain) is currently available on all Unixes. They are also ported to Windows (by Cygwin, MinGW and MinGW-W64). GCC is also a cross-compiler, for producing executables on different platform.

The original GNU C Compiler (GCC) was developed by Richard Stallman, the founder of the GNU Project. Richard Stallman founded the GNU project in 1984 to create a complete Unix-like operating system as free software, to promote freedom and cooperation among computer users and programmers.

Compiling Programs with GCC

GCC is a key component of so-called "**GNU Toolchain**", for developing applications and writing operating systems. The GNU Toolchain includes:

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

GCC is a key component of so-called "**GNU Toolchain**", for developing applications and writing operating systems. The GNU Toolchain includes:

GNU Compiler Collection (GCC): a compiler suite that supports many languages, such as C/C++ and Objective-C/C++.

GNU Make: an automation tool for compiling and building applications.

GNU Binutils: a suite of binary utility tools, including linker and assembler.

GNU Debugger (GDB).

GNU Autotools: A build system including Autoconf, Autoheader, Automake and Libtool.

GNU Bison: a parser generator (similar to lex and yacc).

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

GCC is a key component of so-called "**GNU Toolchain**", for developing applications and writing operating systems. The GNU Toolchain includes:

GNU Compiler Collection (GCC): a compiler suite that supports many languages, such as C/C++ and Objective-C/C++.

GNU Make: an automation tool for compiling and building applications.

GNU Binutils: a suite of binary utility tools, including linker and assembler.

GNU Debugger (GDB).

GNU Autotools: A build system including Autoconf, Autoheader, Automake and Libtool.

GNU Bison: a parser generator (similar to lex and yacc).



We will only use GCC and GDB here.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

To make sure GCC is installed, run the following command in the console (terminal):

```
$ gcc --version
```

Compilation Options

The general GCC compilation command format is:

```
gcc [options] [filenames]
```

with

options: compilation options

filenames: source code file name

Caution: C programs are compiled with `gcc`, whereas C++ programs are compiled using the command `g++`. If using C++, replace the command respectively.

Technically: `g++` is a program that calls GCC and automatically specifies linking against the C++ library.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Example for Using GCC to Compile C Programs

When the program consists of just one source file, we can compile and link our source file `main.c` the following way:

```
$ gcc main.c
```

The compiler will create an executable program.

The default output executable is called `a.out` (Unixes and Mac OS X) or `a.exe` (Windows).

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Example for Using GCC to Compile C Programs

When the program consists of just one source file, we can compile and link our source file `main.c` the following way:

```
$ gcc main.c
```

The compiler will create an executable program.

The default output executable is called `a.out` (Unixes and Mac OS X) or `a.exe` (Windows).

For Unix-based systems:

You need to assign executable file-mode (`x`) to the executable file `a.out`:

```
$ chmod a+x a.out
```

(add executable file-mode `"+x"` to all users `"a+x"`).

We can run the program then by typing `./a.out` in the command line:

```
$ ./a.out
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Example for Using GCC to Compile C Programs

We can also choose to create an executable program with the name we want, by adding the `-o` option to the `gcc` command, placed after the name of the file or files we are compiling, and pressing enter:

```
$ gcc main.c -o "program.out"
```

The complete compiling and running workflow is then:

```
$ gcc -o myprogram.out main.c
$ chmod a+x myprogram.out
$ ./myprogram.out
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Compiling Programs with GCC

There are two methods provided for compiling **multiple source files**.

1. Multiple source files are compiled at the same time. All files need to be recompiled during compilation.

example: Compile test1.c and test2.c and link them to the executable file test.

```
$ gcc test1.c test2.c -o test
```

2. Compile each source file, and then link the target files generated after compilation. During compilation, only modified files need to be recompiled.

example: compile test1.c and test2.c, and link the target files test1.o and test2.o to the executable file test.

```
$ gcc -c test1.c  
$ gcc -c test2.c  
$ gcc test1.o test2.o -o test
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Declaring Variables

To declare (create) a variable, specify the type and assign it a value, we use the following syntax:

```
type variableName = value;
```

with

type: one of C++ types (such as `int`),

variableName: the name of the variable (such as `x` or `myName`).

To create a variable that should store a number, look at the following **example:**

Create a variable called `myNum` of type `int` and assign it the value 15:

```
int myNum = 15;  
cout << myNum;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Declaring Variables

To declare (create) a variable, specify the type and assign it a value, we use the following syntax:

```
type variableName = value;
```

with

type: one of C++ types (such as `int`),

variableName: the name of the variable (such as `x` or `myName`).

To create a variable that should store a number, look at the following **example**:

Create a variable called `myNum` of type `int` and assign it the value 15:

```
int myNum = 15;  
cout << myNum;
```

Note:

`cin` and `cout` are in the C++ header `iostream`. While valid C is (typically) valid C++, this is not true in reverse: `cout` and its counterpart `cin` can't be used in a C program. C uses `printf()` instead.

Declaring Variables

You can also declare a variable without assigning the value, and assign the value later:

example:

```
int myNum;  
myNum = 15;  
cout << myNum;
```

A demonstration of other data types:

```
int myNum = 5;           // Integer (whole number without decimals)  
double myFloatNum = 5.99; // Floating point number (with decimals)  
char myLetter = 'D';     // Character  
string myText = "Hello"; // String (text)  
bool myBoolean = true;   // Boolean (true or false)
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Data Types

The **data type** specifies the size and type of information of a variable.

In C++, there are different types of variables (defined with different keywords), for example:

| Data Type | Size | Description |
|-----------|--------------|---|
| boolean | 1 byte | Stores true or false values |
| char | 1 byte | Stores a single ASCII character |
| int | 2 or 4 bytes | Stores whole numbers, without decimals |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6-7 decimal digits. |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits. |

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Data Types

The **data type** specifies the size and type of information of a variable.

In C++, there are different types of variables (defined with different keywords), for example:

| Data Type | Size | Description |
|-----------|--------------|---|
| boolean | 1 byte | Stores true or false values |
| char | 1 byte | Stores a single ASCII character |
| int | 2 or 4 bytes | Stores whole numbers, without decimals |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6-7 decimal digits. |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits. |

Note:

Some compilers consider also longer integer formats of up to 32 or 64 bit. Using GCC, on 64-bit architectures, long int is at least 64 bit. On 32-bit, long int is at least 32 bit.

C++ Data Types

The `char` data type is used to store a single character. The character must be surrounded by single quotes, like `'A'` or `'c'`:

example:

```
char myGrade = 'B';  
cout << myGrade;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Data Types

The `char` data type is used to store a single character. The character must be surrounded by single quotes, like `'A'` or `'c'`:

example:

```
char myGrade = 'B';  
cout << myGrade;
```

The `string` data type is used to store a sequence of characters. String values must be surrounded by double quotes. To use strings, you must include an additional header file in the source code, the `<string>` library.

example:

```
// Include the string library  
#include <string>  
  
// Create a string variable  
string greeting = "Hello";  
  
// Output string value  
cout << greeting;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Outputting Variables

The `cout` object is used together with the `<<` operator to output (print) variables. To combine both text and a variable, separate them with the `<<` operator.

example:

```
int myAge = 35;  
cout << "I am " << myAge << " years old.";
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Outputting Variables

The `cout` object is used together with the `<<` operator to output (print) variables. To combine both text and a variable, separate them with the `<<` operator.

example:

```
int myAge = 35;  
cout << "I am " << myAge << " years old.";
```

example: declaring more than one variable, and displaying the sum

```
int x = 5, y = 6, z = 50;  
cout << x + y + z;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Constants

When you do not want others (or yourself) to change existing variable values, use the `const` keyword (this will declare the variable as "constant", which means unchangeable and read-only).

example:

```
const int myNum = 15; // myNum will always be 15  
myNum = 10; // error: assignment of read-only variable 'myNum'
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Constants

When you do not want others (or yourself) to change existing variable values, use the `const` keyword (this will declare the variable as "constant", which means unchangeable and read-only).

example:

```
const int myNum = 15; // myNum will always be 15  
myNum = 10; // error: assignment of read-only variable 'myNum'
```

You should always declare the variable as constant when you have values that are unlikely to change:

example:

```
const int minutesPerHour = 60;  
const float PI = 3.14;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Constants

Note:

When you declare a constant variable, it must be assigned with a value:

example:

```
const int minutesPerHour = 60;
```

This however will not work:

```
const int minutesPerHour;  
minutesPerHour = 60; // error
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ User Input

We have already seen that `cout` is used to output (print) values. Now we will use `cin` to get user input.

`cin` is a predefined variable that reads data from the keyboard with the extraction operator `>>`.

In the following **example**, the user can input a number, which is stored in the variable `x`. Then we print the value of `x`:

```
int x;  
cout << "Type a number: "; // Type a number and press enter  
cin >> x; // Get user input from the keyboard  
cout << "Your number is: " << x; // Display the input value
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ User Input

example: Creating a Simple Calculator

In this example, the user must input two numbers. Then we print the sum by calculating (adding) the two numbers:

```
int x, y;  
int sum;  
cout << "Type a number: ";  
cin >> x;  
cout << "Type another number: ";  
cin >> y;  
sum = x + y;  
cout << "Sum is: " << sum;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ User Input

Operators are used to perform operations on variables and values.

They work in an intuitive way. In the **example** below, we use the `+` operator to add together two values:

```
int x = 100 + 50;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

| Operator | Name | Description | Example |
|----------|----------------|--|----------|
| + | Addition | Adds together two values | $x + y$ |
| - | Subtraction | Subtracts one value from another | $x - y$ |
| * | Multiplication | Multiplies two values | $x * y$ |
| / | Division | Divides one value by another | x / y |
| % | Modulus | Returns the division remainder | $x \% y$ |
| ++ | Increment | Increases the value of a variable by 1 | $++x$ |
| -- | Decrement | Decreases the value of a variable by 1 | $--x$ |

Motivation

What is C/C++?

Getting Started

Compiling Programs with GCC

Variables

C++ Operators

Summary & Outlook

Assignment Operators

In addition, C++ uses the following **assignment operators**:

| Operator | Example | Same As |
|----------|---------|------------|
| = | x = 5 | |
| += | x += 3 | x = x + 3 |
| -= | x -= 3 | x = x - 3 |
| *= | x *= 3 | x = x * 3 |
| /= | x /= 3 | x = x / 3 |
| %= | x %= 3 | x = x % 3 |
| &= | x &= 3 | x = x & 3 |
| = | x = 3 | x = x 3 |
| ^= | x ^= 3 | x = x ^ 3 |
| >>= | x >>= 3 | x = x >> 3 |
| <<= | x <<= 3 | x = x << 3 |

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Comparison Operators

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to make decisions.

The return value of a comparison is either 1 or 0, which means true (1) or false (0). These values are known as Boolean values.

In the following **example**, we use the greater than operator ($>$) to find out if 5 is greater than 3:

```
int x = 5;  
int y = 3;  
cout << (x > y); // returns 1 (true) because 5 is greater than 3
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Comparison Operators

A list of all comparison operators in C++:

| Operator | Name | Example |
|--------------------|--------------------------|------------------------|
| <code>==</code> | Equal to | <code>x == y</code> |
| <code>!=</code> | Not equal | <code>x != y</code> |
| <code>></code> | Greater than | <code>x > y</code> |
| <code><</code> | Less than | <code>x < y</code> |
| <code>>=</code> | Greater than or equal to | <code>x >= y</code> |
| <code><=</code> | Less than or equal to | <code>x <= y</code> |

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Logical Operators

As with comparison operators, you can also test for true (1) or false (0) values with **logical operators**:

| Operator | Name | Description | Example |
|-------------------------|-------------|--|---|
| <code>&&</code> | Logical and | Returns true if both statements are true | <code>x < 5 && x < 10</code> |
| <code> </code> | Logical or | Logical or Returns true if one of the statements is true | <code>x < 5 x < 4</code> |
| <code>!</code> | Logical not | Reverse the result, returns false if the result is true | <code>!(x < 5 && x < 10)</code> |

Motivation

What is C/C++?

Getting Started

Compiling Programs with GCC

Variables

C++ Operators

Summary & Outlook

C++ <cmath> Library

Other functions, such as `sqrt` (square root), `round` (rounds a number) and `log` (natural logarithm), can be found in the `<cmath>` header file.

example:

```
// Include the cmath library  
#include <cmath>  
  
cout << sqrt(64)<<endl;  
cout << round(2.6)<<endl;  
cout << log(2)<<endl;
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Control Flow Statements

Control flow statements are necessary to make a program react to user input as well to the results of computations.

The typical control flow statements can be categorized by their effect:

- Continuation at a different statement (unconditional branch or jump)
- Executing a set of statements only if some condition is met (choice - i.e., conditional branch)
- Executing a set of statements zero or more times, until some condition is met (i.e., loop - the same as conditional branch)
- Executing a set of distant statements, after which the flow of control usually returns (subroutines, coroutines, and continuations)
- Stopping the program, preventing any further execution (unconditional halt)

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Conditions and If Statements

We have seen that C++ supports the usual **logical conditions**:

Less than: $a < b$

Less than or equal to: $a \leq b$

Greater than: $a > b$

Greater than or equal to: $a \geq b$

Equal to: $a == b$

Not Equal to: $a != b$

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Conditions and If Statements

We have seen that C++ supports the usual **logical conditions**:

Less than: $a < b$

Less than or equal to: $a \leq b$

Greater than: $a > b$

Greater than or equal to: $a \geq b$

Equal to: $a == b$

Not Equal to: $a != b$



We can use these conditions to perform different actions for different outcomes of those decisions.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Conditions and If Statements

C++ has the following conditional statements:

- **if**: to specify a block of code to be executed, if a specified condition is true
- **else**: to specify a block of code to be executed, if the same condition is false
- **else if**: to specify a new condition to test, if the first condition is false
- **switch**: to specify many alternative blocks of code to be executed

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

The if Statement


Use the `if` statement to specify a block of C++ code to be executed if a condition is true.

Syntax:

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```


Condition is true

```
int number = 5;  
  
if (number > 0) {  
    // code  
}  
  
// code after if
```



Condition is false

```
int number = 5;  
  
if (number < 0) {  
    // code  
}  
  
// code after if
```



Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

The if Statement

Note that `if` is in lowercase letters. Uppercase letters (`If` or `IF`) will generate an error.

In the **example** below, we test two values to find out if 20 is greater than 18. If the condition is true, print some text:

```
if (20 > 18) {  
    cout << "20 is greater than 18";  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

The else Statement

Use the `else` and `else if` statement to specify a new condition if the first condition is false. There can be multiple `else if` statements.

Syntax:

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false  
    // and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false  
    // and condition2 is false  
}
```

1st Condition is true

```
int number = 2;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
//code after if
```

2nd Condition is true

```
int number = 0;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
//code after if
```

All Conditions are false

```
int number = -2;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
//code after if
```

C++ switch Statements

Use the `switch` statement to select one of many code blocks to be executed.

Syntax:

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

This is how it works:

The `switch` expression is evaluated once.

The value of the expression is compared with the values of each case.

If there is a match, the associated block of code is executed.

The `break` and `default` keywords are optional.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

The break Keyword

When the program reaches a `break` keyword, it breaks out of the `switch` block. This saves execution time because it "ignores" the execution of all the rest of the code in the `switch` block. See the following **example**:

```
int day = 4;
switch (day) {
    case 1:
        cout << "Monday";
        break;
    case 2:
        cout << "Tuesday";
        break;
    case 3:
        cout << "Wednesday";
        break;
    case 4:
        cout << "Thursday";
        break;
    case 5:
        cout << "Friday";
        break;
    case 6:
        cout << "Saturday";
        break;
    case 7:
        cout << "Sunday";
        break;
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ Loops

Loops can execute a block of code as long as a specified condition holds. Loops are handy because they save time, reduce errors, and they make code more readable.

Especially when dealing with large amounts of data that must be processed automatically they are essential.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ while Loop

The `while` loop loops through a block of code as long as a specified condition is true.

Syntax:

```
while (condition) {  
    // code block to be executed  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ while Loop

The **while** loop loops through a block of code as long as a specified condition is true.

Syntax:

```
while (condition) {  
    // code block to be executed  
}
```

In the **example** below, the code in the loop will run as long as the variable **i** is less than 5:

```
int i = 0;  
while (i < 5) {  
    cout << i << "\n";  
    i++;  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ do ... while Loop

The do ... while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

```
do {  
    // code block to be executed  
}  
while (condition);
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ do ... while Loop

The do ... while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

```
do {  
    // code block to be executed  
}  
while (condition);
```

The **example** below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested.

```
int i = 0;  
do {  
    cout << i << "\n";  
    i++;  
}  
while (i < 5);
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ for Loop

When you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop.

Syntax:

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

how it works:

Statement 1 is executed (one time) before the execution of the code block.
Statement 2 defines the condition for executing the code block.
Statement 3 is executed (every time) after the code block has been executed.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ for Loop

The **example** below will print the numbers 0 to 4:

```
for (int i = 0; i < 5; i++) {  
    cout << i << "\n";  
}
```

how it works:

Statement 1 sets a variable before the loop starts (`int i = 0`).

Statement 2 defines the condition for the loop to run (`i` must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.

Statement 3 increases a value (`i++`) each time the code block in the loop has been executed.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ for Loop

The **example** below will print the numbers 0 to 4:

```
for (int i = 0; i < 5; i++) {  
    cout << i << "\n";  
}
```

how it works:

Statement 1 sets a variable before the loop starts (`int i = 0`).

Statement 2 defines the condition for the loop to run (`i` must be less than 5). If the condition is true, the loop will start over again, if it is false, the loop will end.

Statement 3 increases a value (`i++`) each time the code block in the loop has been executed.

This **example** will only print even values between 0 and 10:

```
for (int i = 0; i <= 10; i = i + 2) {  
    cout << i << "\n";  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ for-each Loop

There is also a "for-each loop" which is used exclusively to loop through elements in an array (or other data sets).

Syntax:

```
for (type variableName : arrayName) {  
    // code block to be executed  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ for-each Loop

There is also a "for-each loop" which is used exclusively to loop through elements in an array (or other data sets).

Syntax:

```
for (type variableName : arrayName) {  
    // code block to be executed  
}
```

The following **example** outputs all elements in an array, using a "for-each loop":

```
int myNumbers[5] = {10, 20, 30, 40, 50};  
for (int i : myNumbers) {  
    cout << i << "\n";  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ break in loops

We have already seen the `break` statement used in an earlier chapter of this tutorial. It was used to "jump out" of a `switch` statement.

The `break` statement can also be used to jump out of a loop.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ break in loops

We have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

This **example** jumps out of the loop when i is equal to 4:

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        break;  
    }  
    cout << i << "\n";  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ break in loops

We have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

This **example** jumps out of the loop when i is equal to 4:

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        break;  
    }  
    cout << i << "\n";  
}
```

Caution: Using the break statement can make code hard to read (commonly known as "spaghetti code"). It is recommended to restrict it to cases where writing the code differently would make it even more complicated.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ continue

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This **example** skips the value of 4:

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        continue;  
    }  
    cout << i << "\n";  
}
```

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

C++ continue

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This **example** skips the value of 4:

```
for (int i = 0; i < 10; i++) {  
    if (i == 4) {  
        continue;  
    }  
    cout << i << "\n";  
}
```

Caution: Also the `continue` statement can make code hard to read, so the same recommendation applies as for the `break` statement.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Summary: Introduction to C/C++

C++ finds varied usage in applications such as:

- Operating Systems & Systems Programming. e.g. Linux-based OS (Ubuntu etc.)
- Browsers (Chrome & Firefox)
- Graphics & Game engines (Photoshop, Blender, Unreal-Engine)
- Database Engines (MySQL, MongoDB, Redis etc.)
- Cloud/Distributed Systems

In **astronomy**, C++ knowledge can be very valuable for speeding up code that otherwise might be written in Python.

It can also be useful to understand so-called legacy code (= old code that needs to be maintained or rewritten).

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

Summary: Introduction to C/C++

Reference Books:

The C++ Programming Language by Bjarne Stroustrup

Effective C++: 55 Specific Ways to Improve Your Programs and Designs
by Scott Meyers

Data Structures and Algorithm Analysis in C++ by Mark Allen Weiss

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook

An Outlook: Introduction to C/C++ (II)

In the next lecture, we will build on what we have learned here to write more complex programs that can include such as **functions** and use the concepts of **references** and **pointers**.

We will also see how to **debug** programs.

Motivation

What is
C/C++?

Getting
Started

Compiling
Programs with
GCC

Variables

C++
Operators

Summary &
Outlook