Fundamental Programming

Week 8: Basic Concepts of the C Programming Language

Dr. Maria Irmina Prasetiyowati, S.Kom,. M.T.
Alethea Suryadibrata, S.Kom., M.Eng.
Putri Sanggabuana Setiawan, S.Kom, M.T.I.
Januar Wahjudi, S.Kom., M.Sc.
Drs Slamet Aji Pamungkas, M.Eng
Kursehi Falgenti S.Kom., M.Kom.

Weekly Learning Outcomes for Subjects (Sub-CPMK):

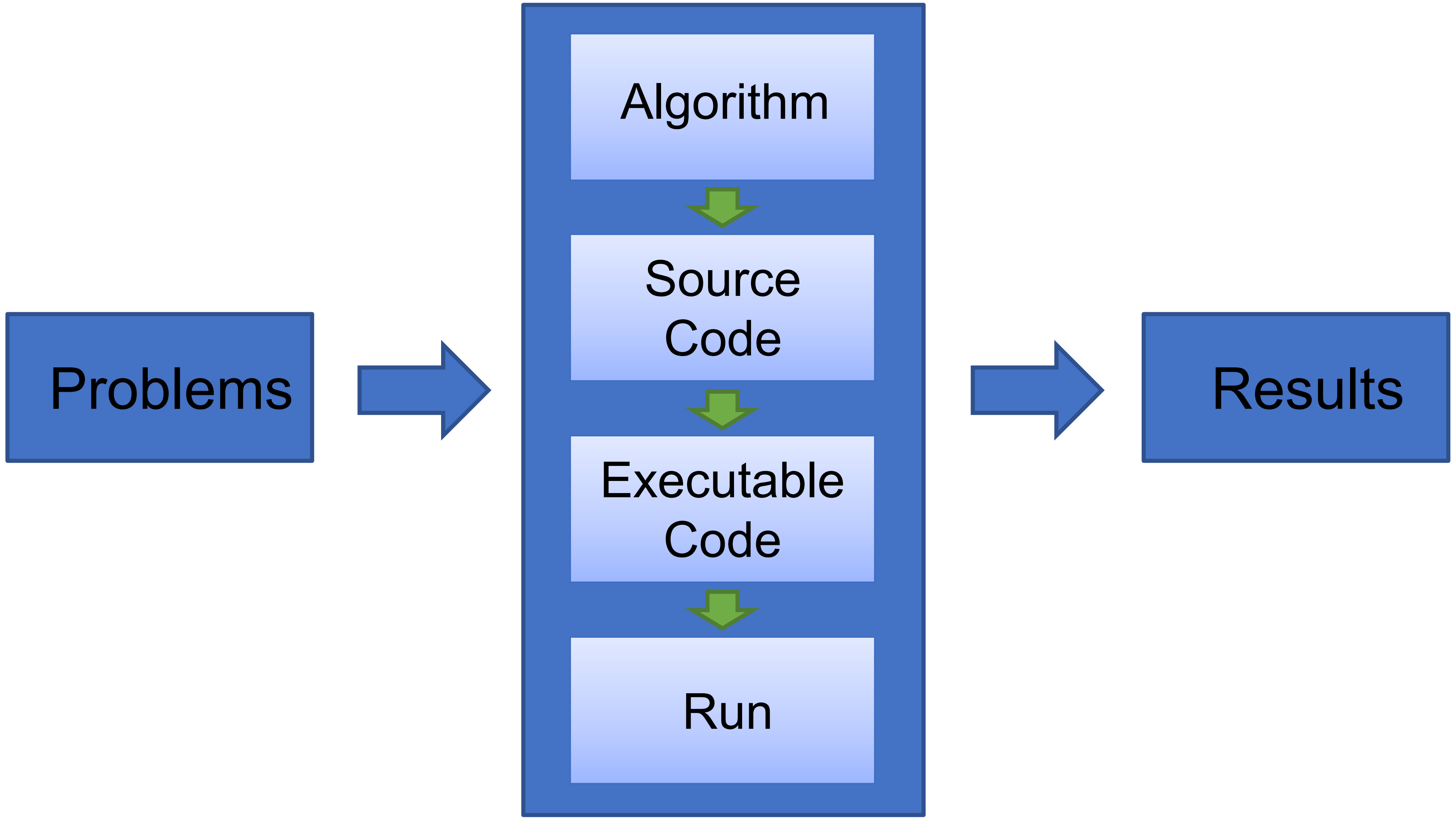
Sub-CPMK 0212: Students are able to explain the basic concepts of C language programming (C2).

Outline

1. Programming Languages
2. Integrated Development Environment
3. Coding
   - Basic Structure of C
   - Escape Sequences
   - Code Commenting
4. Data type
5. Contranta
6. Variable

PROGRAMMING LANGUAGES

Problems → Solutions → Results



Algorithm

- A procedure for solving a problem in terms of
  1. the **actions** to be executed
  2. The **order** in which these actions are to be executed

Algorithm

- Criteria
  1. Finiteness
     - An algorithm must terminate after a finite number of steps
  2. Definiteness
     - Each step of an algorithm must be precisely defined
  3. Input
     - rather useless
     - An algorithm has zero or more inputs
     - Input values are supplied either before the algorithm starts or as the algorithm runs
  4. Output
     - An algorithm has one or more outputs
     - Output values are specifically determined by the inputs
  5. Effectiveness
     - An algorithm is supposed to be effective
     - Its operations must be able to be done exactly and in a finite length of time

Programming languages

- A formal constructed languages designed to communicate instructions to a machine
- Three General Types of Programming Languages
  1. Machine Languages
  2. Assembly Languages
  3. High-Level Languages

Programming languages

1. Machine Language
   - The “natural language” of a computer
   - Defined by its hardware design
   - Strings of numbers that computers could directly understand

```
8B542408    83FA0077    06B80000    0000C383
FA027706    B8010000    00C353BB    01000000
C9010000    008D0419    83FA0376    078BD98B
B84AEBF1    5BC3
```

Programming languages

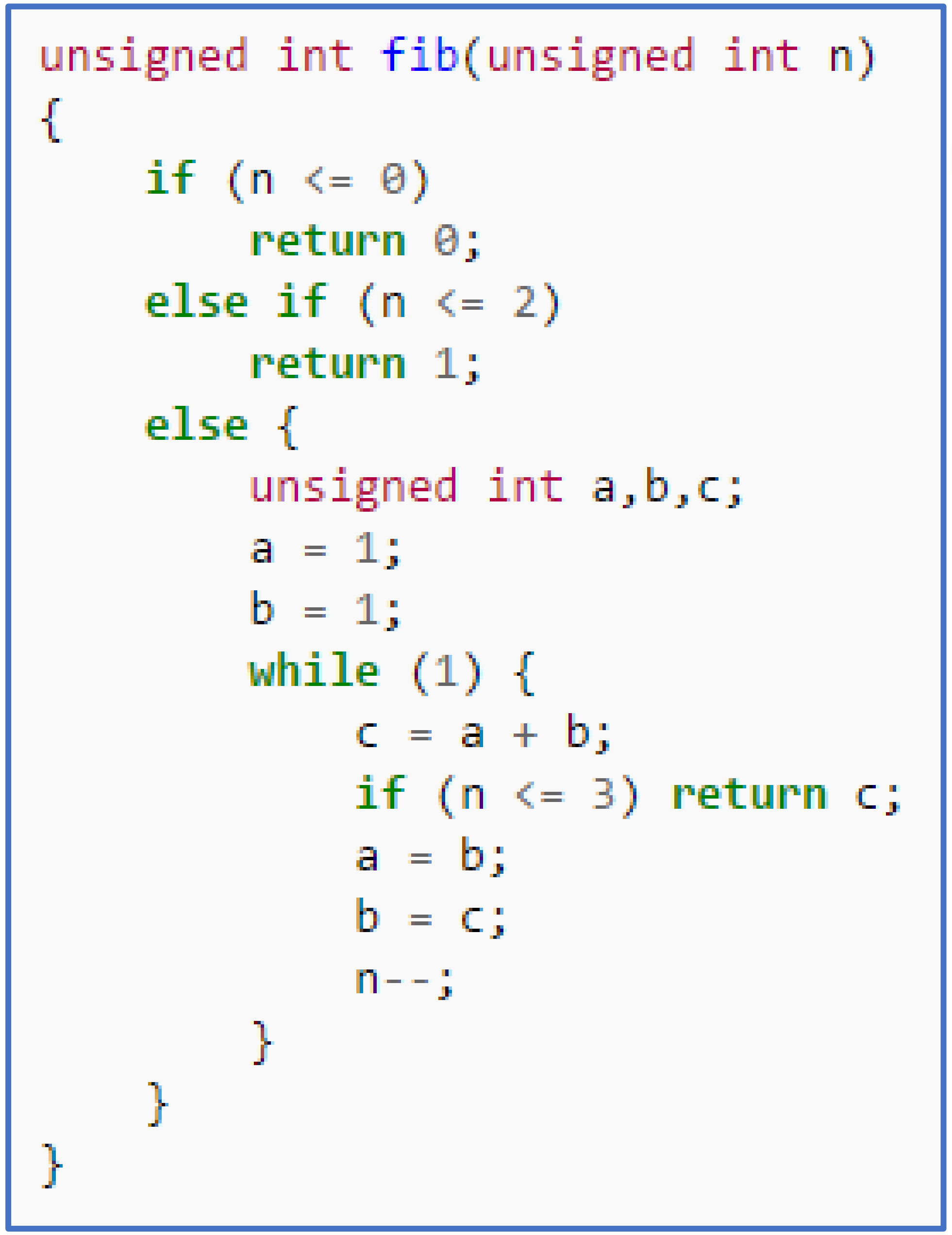
2. Assembly Language
   - English-like abbreviations to represent elementary operations

```
fib:
    mov edx, [esp+8]
    cmp edx, 0
    ja @f
    mov eax, 0
    ret

    @@:
    cmp edx, 2
    ja @f
    mov eax, 1
    ret

    @@:
    push ebx
    mov ebx, 1
    mov ecx, 1
```

```
    @@:
        lea eax, [ebx+ecx]
        cmp edx, 3
        jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
    jmp @b

    @@:
    pop ebx
    ret
```

Programming languages

3. High-Level Language
   - Single statements could be written to accomplish substantial tasks
   - Allow programmers to write instructions that look almost like everyday English and contain commonly used mathematical notations

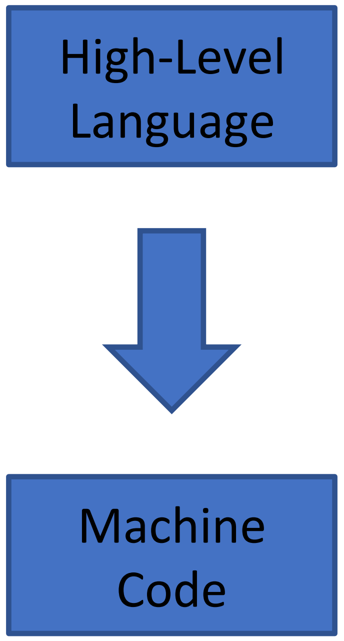


```
unsigned int fib(unsigned int n)
{
    if (n <= 0)
        return 0;
    else if (n <= 2)
        return 1;
    else {
        unsigned int a,b,c;
        a = 1;
        b = 1;
        while (1) {
            c = a + b;
            if (n <= 3) return c;
            a = b;
            b = c;
            n--;
        }
    }
}
```

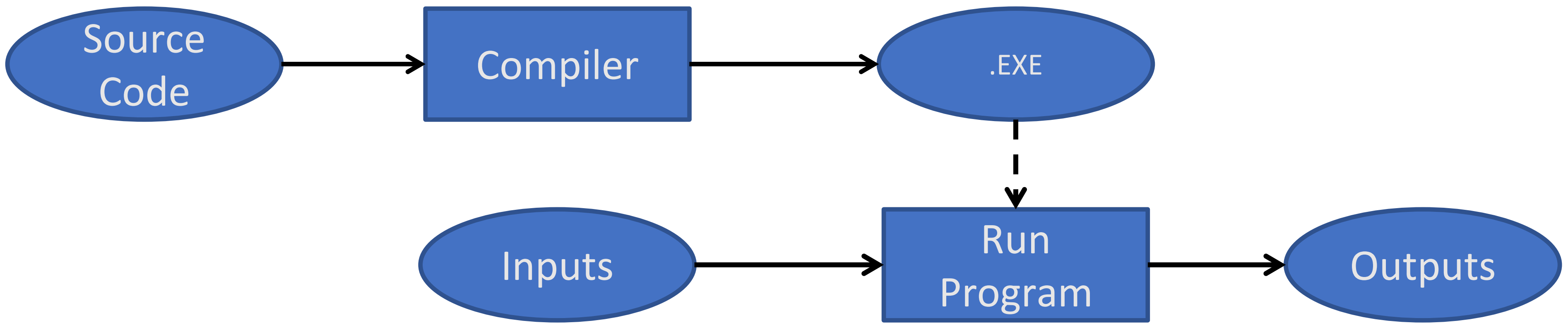
Compiler vs Interpreter

Compiler	Interpreter
Compiler takes **entire** program as input	Interpreter takes **single** instruction as input
Intermediate object code is **generated**	**No** intermediate object code is generated

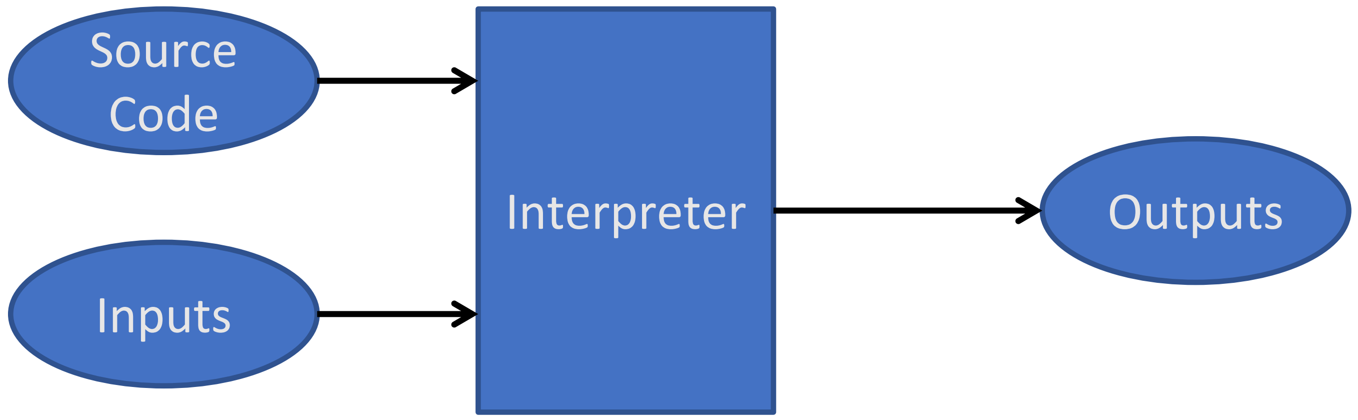
- The process of compiling a high-level language program into machine language can take a considerable amount of computer time
- Interpreter programs execute high-level language programs directly (without the delay of compilation), although slower than compiled programs run



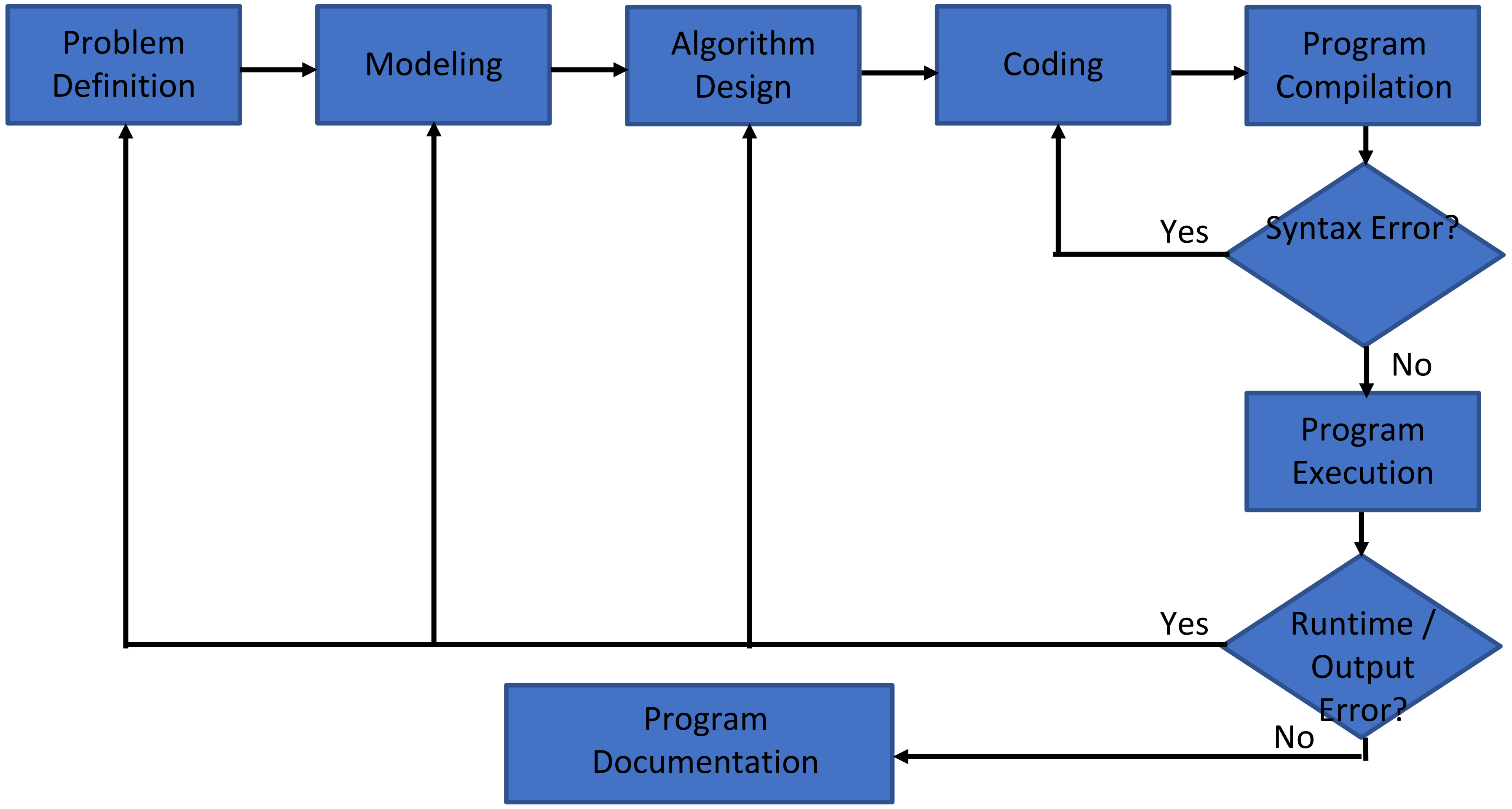
Compiler



Interpreter



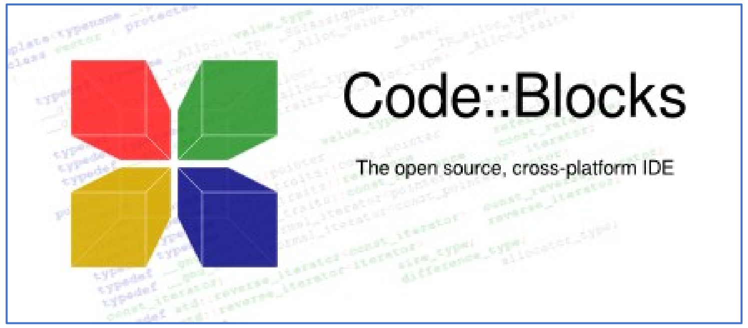
Program development cycle

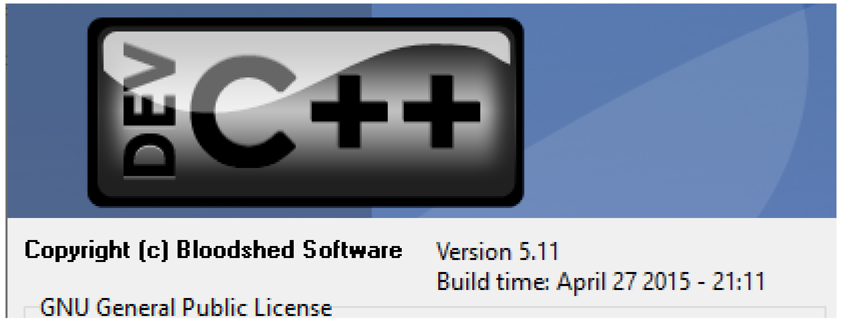


INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

Integrated development environment (ide)

- Most high-level language compilers are sold as part of an Integrated Development Environment (IDE)
- IDE is a package that combines a simple word processor with a compiler, linker, loader, and tools for finding errors





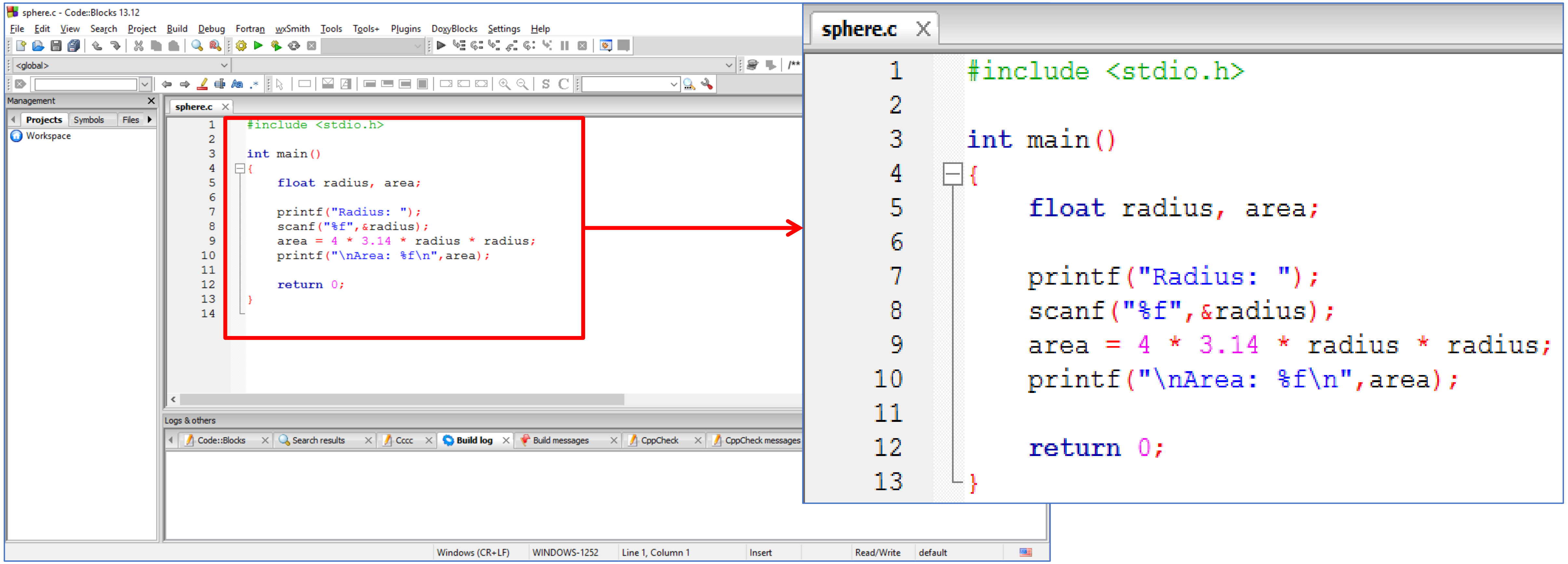
Copyright (c) Bloodshed Software
Version 5.11
Build time: April 27 2015 - 21:11
GNU General Public License

Code::Blocks

- **Code::Blocks** is a free C, C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.
- Website: http://www.codeblocks.org/
- Download link: http://www.codeblocks.org/downloads/26
- Recommended Installer for Windows Operating Systems: **codeblocks-20.03mingw-setup.exe** → Code::Blocks installer with additional GCC/G++/GFortran compiler and GDB debugger from MinGW-W64 project

Coding

1. Calculate the surface area of a sphere

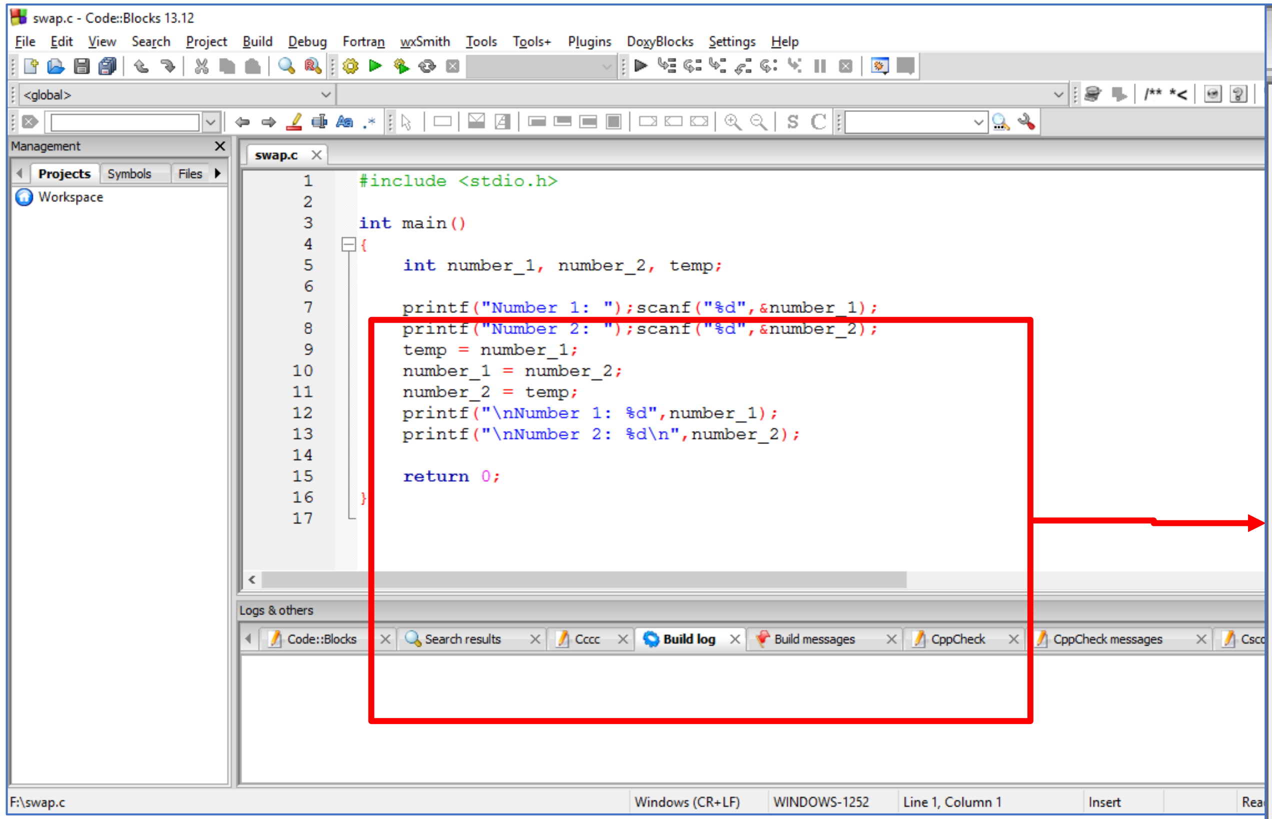


```c
#include <stdio.h>

int main()
{
    float radius, area;

    printf("Radius: ");
    scanf("%f",&radius);
    area = 4 * 3.14 * radius * radius;
    printf("\nArea: %f\n",area);

    return 0;
}
```

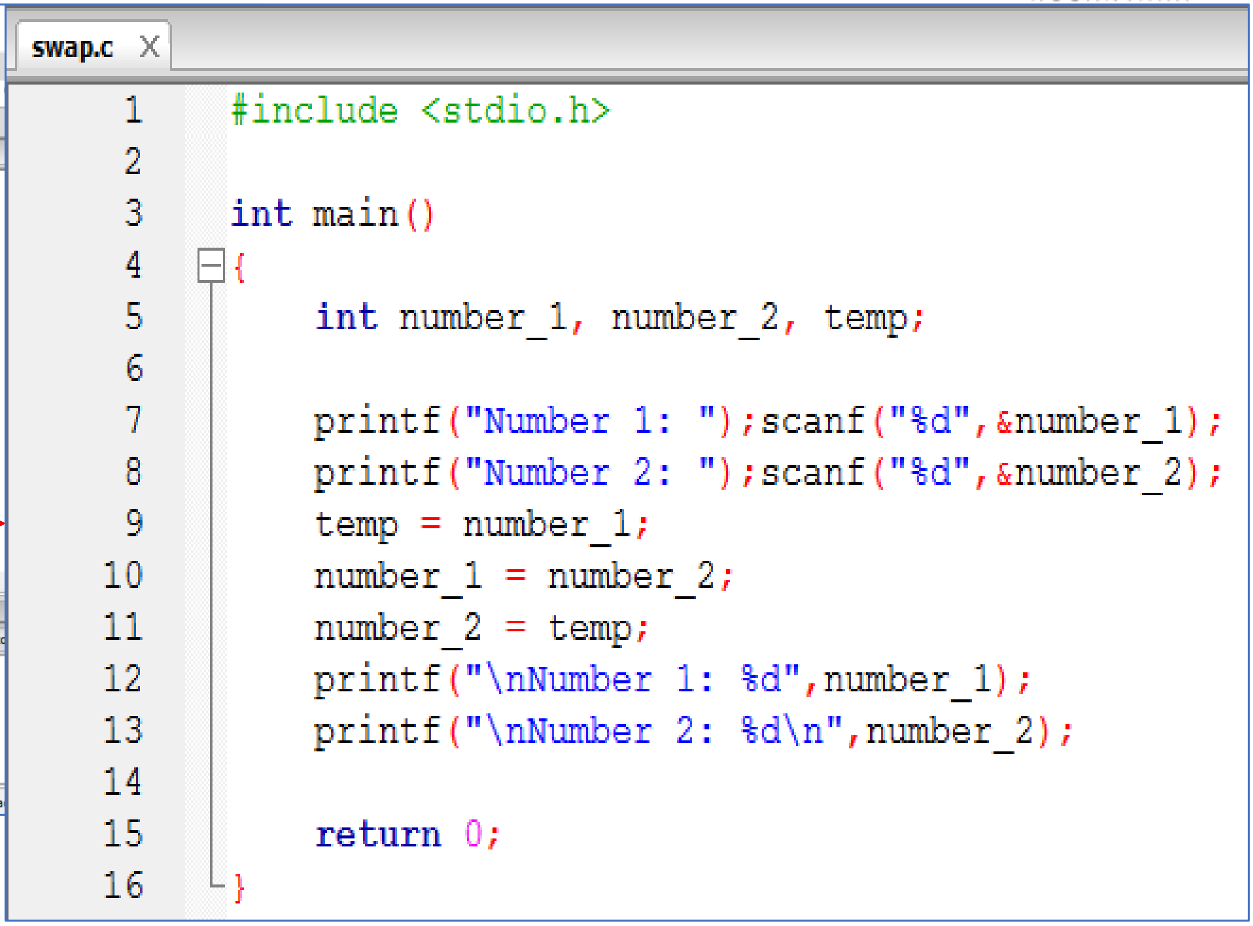
sphere.c - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck CppCheck messages
Windows (CR+LF) WINDOWS-1252 Line 1, Column 1 Insert Read/Write default

Coding

2. Swap two numbers



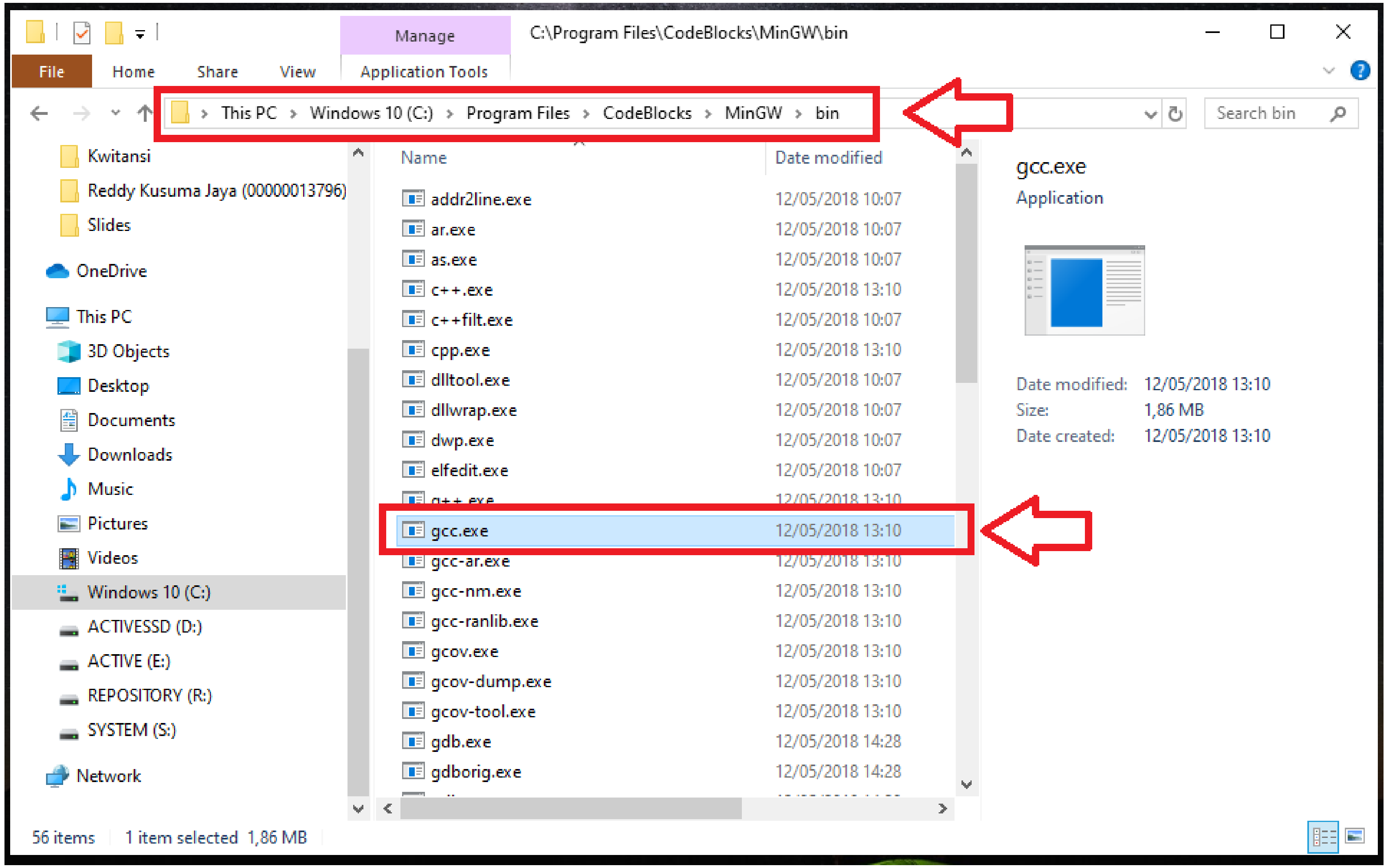
```c
#include <stdio.h>

int main()
{
    int number_1, number_2, temp;

    printf("Number 1: ");scanf("%d",&number_1);
    printf("Number 2: ");scanf("%d",&number_2);
    temp = number_1;
    number_1 = number_2;
    number_2 = temp;
    printf("\nNumber 1: %d",number_1);
    printf("\nNumber 2: %d\n",number_2);

    return 0;
}
```



```c
#include <stdio.h>

int main()
{
    int number_1, number_2, temp;

    printf("Number 1: ");scanf("%d",&number_1);
    printf("Number 2: ");scanf("%d",&number_2);
    temp = number_1;
    number_1 = number_2;
    number_2 = temp;
    printf("\nNumber 1: %d",number_1);
    printf("\nNumber 2: %d\n",number_2);

    return 0;
}
```

Program compilation (compile)

- Compile Independently

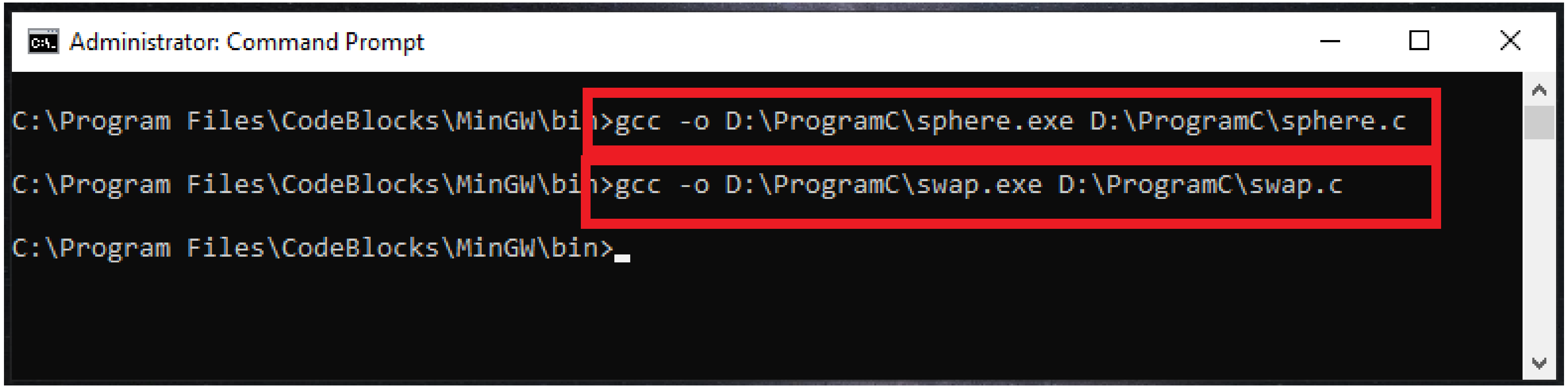


Name	Date modified
addr2line.exe	12/05/2018 10:07
ar.exe	12/05/2018 10:07
as.exe	12/05/2018 10:07
c++.exe	12/05/2018 13:10
c++filt.exe	12/05/2018 10:07
cpp.exe	12/05/2018 13:10
dlltool.exe	12/05/2018 10:07
dllwrap.exe	12/05/2018 10:07
dwp.exe	12/05/2018 10:07
elfedit.exe	12/05/2018 10:07
g++.exe	12/05/2018 13:10
gcc.exe	12/05/2018 13:10
gcc-ar.exe	12/05/2018 13:10
gcc-nm.exe	12/05/2018 13:10
gcc-ranlib.exe	12/05/2018 13:10
gcov.exe	12/05/2018 13:10
gcov-dump.exe	12/05/2018 13:10
gcov-tool.exe	12/05/2018 13:10
gdb.exe	12/05/2018 14:28
gdborig.exe	12/05/2018 14:28

Date modified: 12/05/2018 13:10
Size: 1,86 MB
Date created: 12/05/2018 13:10

Program compilation (compile)

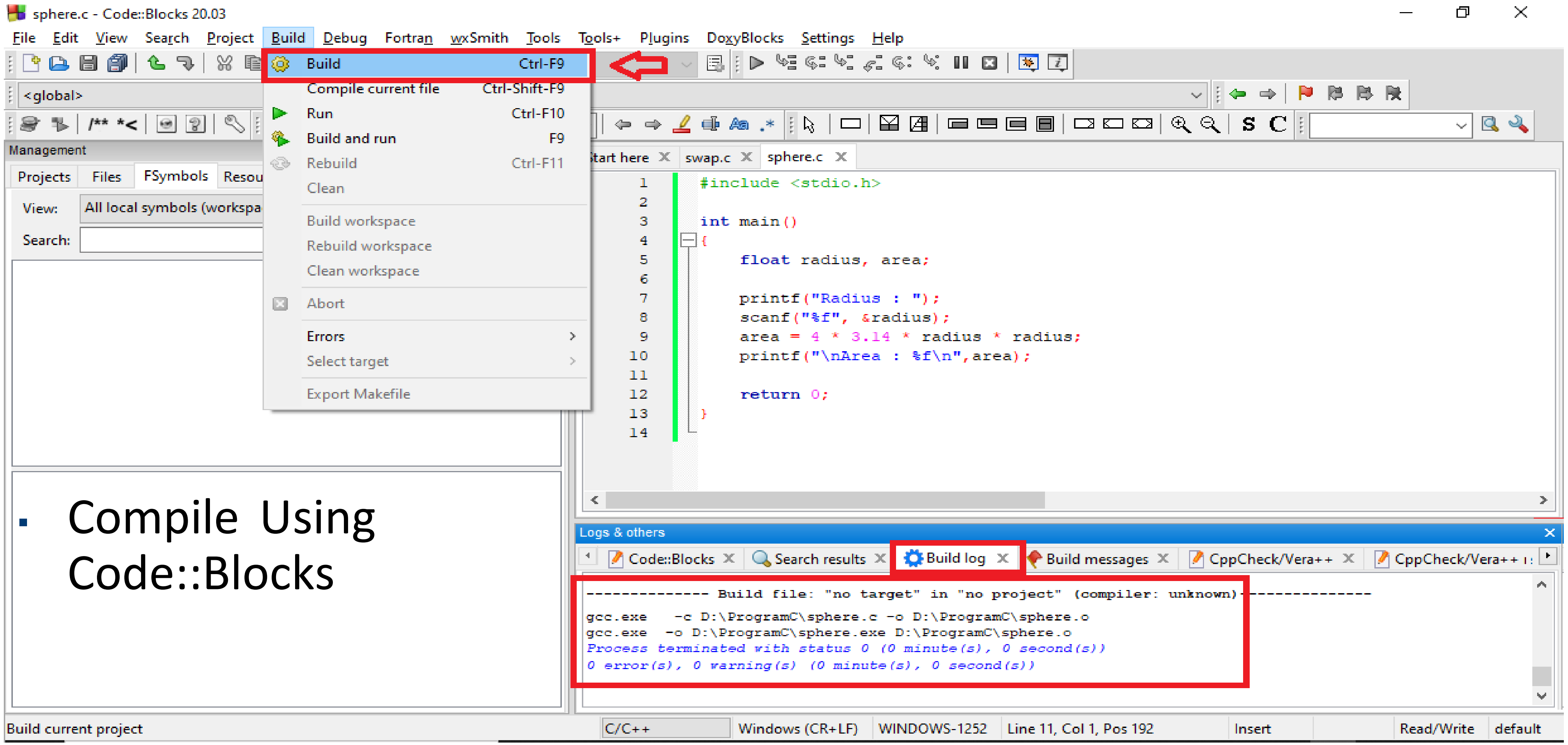
- Compile Independently
  - Command: gcc -o [output_file].exe [source_file].c



Administrator: Command Prompt

```
C:\Program Files\CodeBlocks\MinGW\bin>gcc -o D:\ProgramC\sphere.exe D:\ProgramC\sphere.c

C:\Program Files\CodeBlocks\MinGW\bin>gcc -o D:\ProgramC\swap.exe D:\ProgramC\swap.c

C:\Program Files\CodeBlocks\MinGW\bin>_
```

Program compilation (compile)


sphere.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Build Ctrl-F9
Compile current file Ctrl-Shift-F9
Run Ctrl-F10
Build and run F9
Rebuild Ctrl-F11
Clean
Build workspace
Rebuild workspace
Clean workspace
Abort
Errors
Select target
Export Makefile
<global>
Management
Projects Files FSymbols Resou
View: All local symbols (workspa
Search:
Start here swap.c sphere.c
Logs & others
Code::Blocks Search results Build log Build messages CppCheck/Vera++ CppCheck/Vera++
Build current project
C/C++ Windows (CR+LF) WINDOWS-1252 Line 11, Col 1, Pos 192 Insert Read/Write default

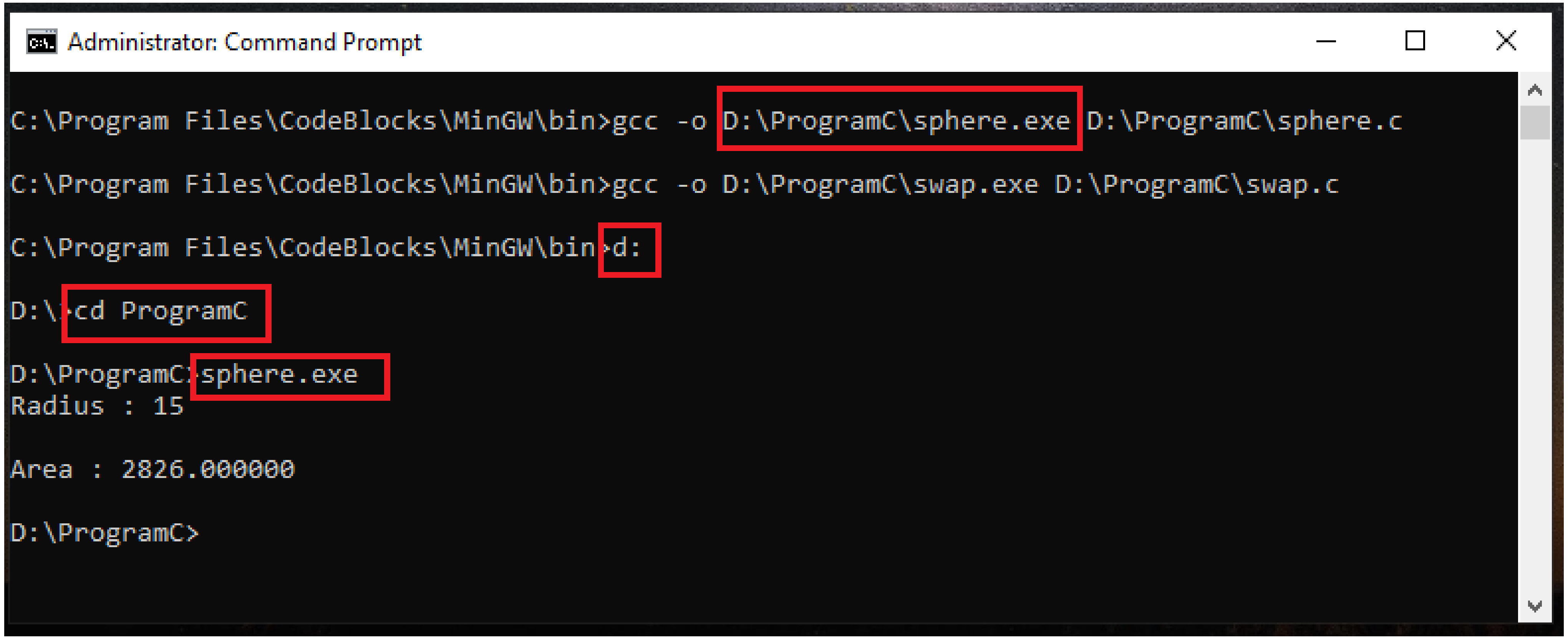
```c
#include <stdio.h>

int main()
{
    float radius, area;

    printf("Radius : ");
    scanf("%f", &radius);
    area = 4 * 3.14 * radius * radius;
    printf("\nArea : %f\n",area);

    return 0;
}
```

```
-------------- Build file: "no target" in "no project" (compiler: unknown)---------------

gcc.exe   -c D:\ProgramC\sphere.c -o D:\ProgramC\sphere.o
gcc.exe  -o D:\ProgramC\sphere.exe D:\ProgramC\sphere.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
```

- Compile Using Code::Blocks

Program execution (run)

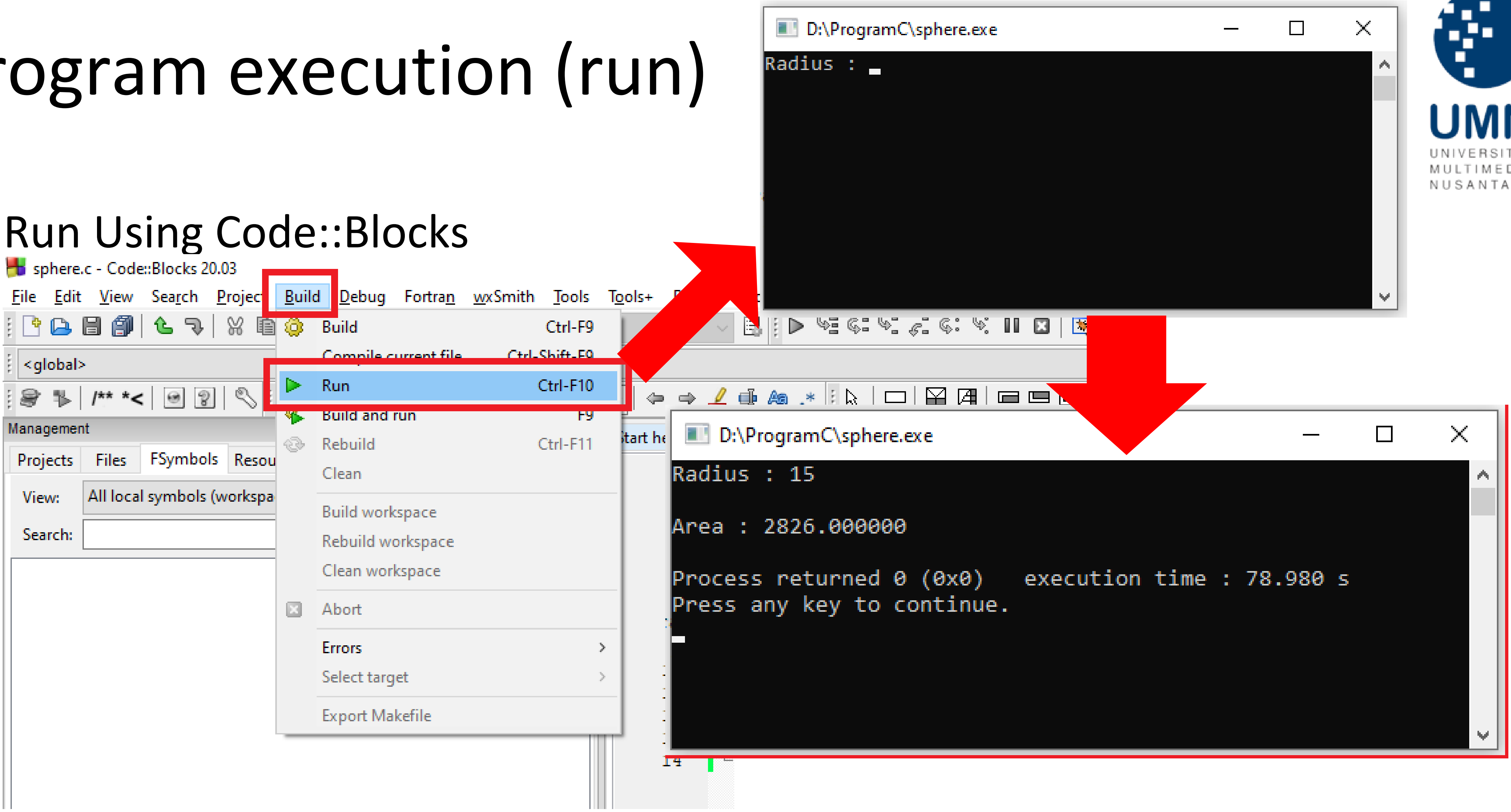
- Run Using Command Prompt



```
C:\Program Files\CodeBlocks\MinGW\bin>gcc -o D:\ProgramC\sphere.exe D:\ProgramC\sphere.c

C:\Program Files\CodeBlocks\MinGW\bin>gcc -o D:\ProgramC\swap.exe D:\ProgramC\swap.c

C:\Program Files\CodeBlocks\MinGW\bin>d:

D:\>cd ProgramC

D:\ProgramC>sphere.exe
Radius : 15

Area : 2826.000000

D:\ProgramC>
```

Program execution (run)

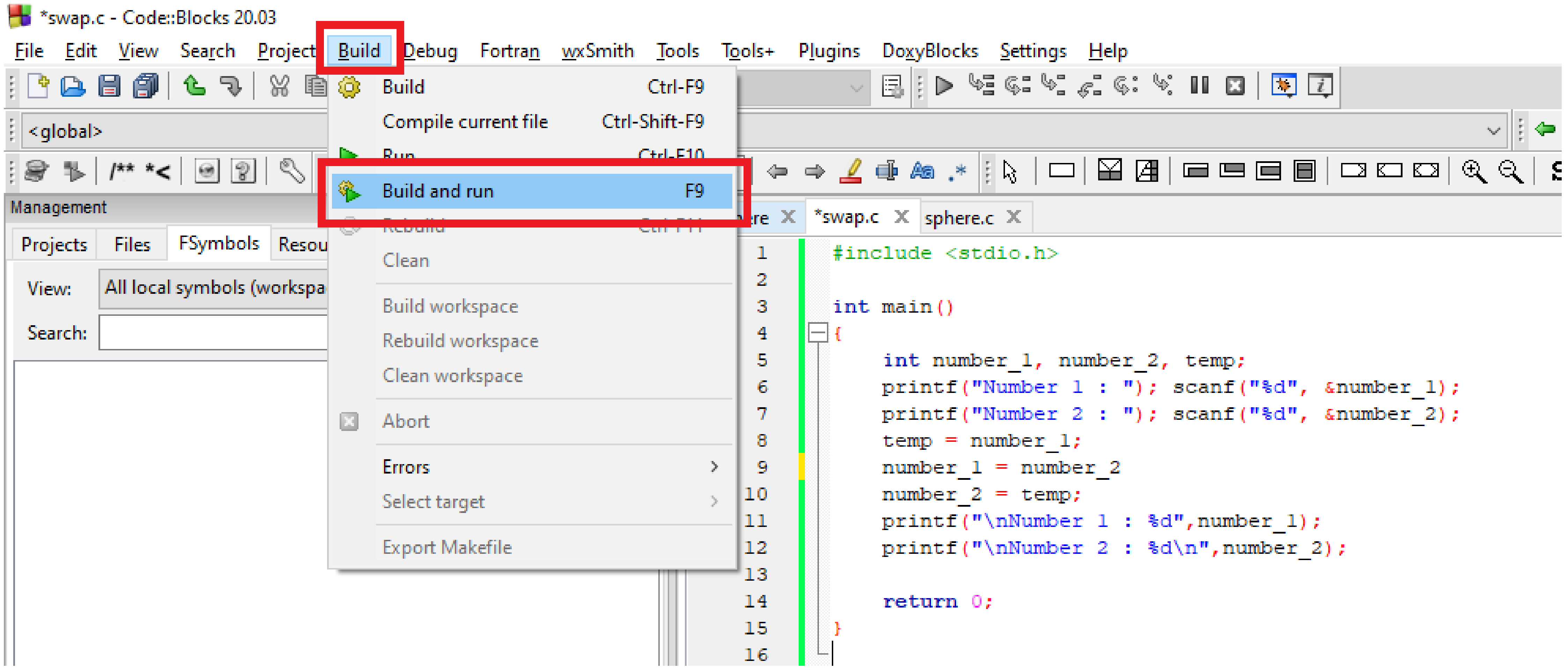
- Run Using Code::Blocks



```
Radius :
```

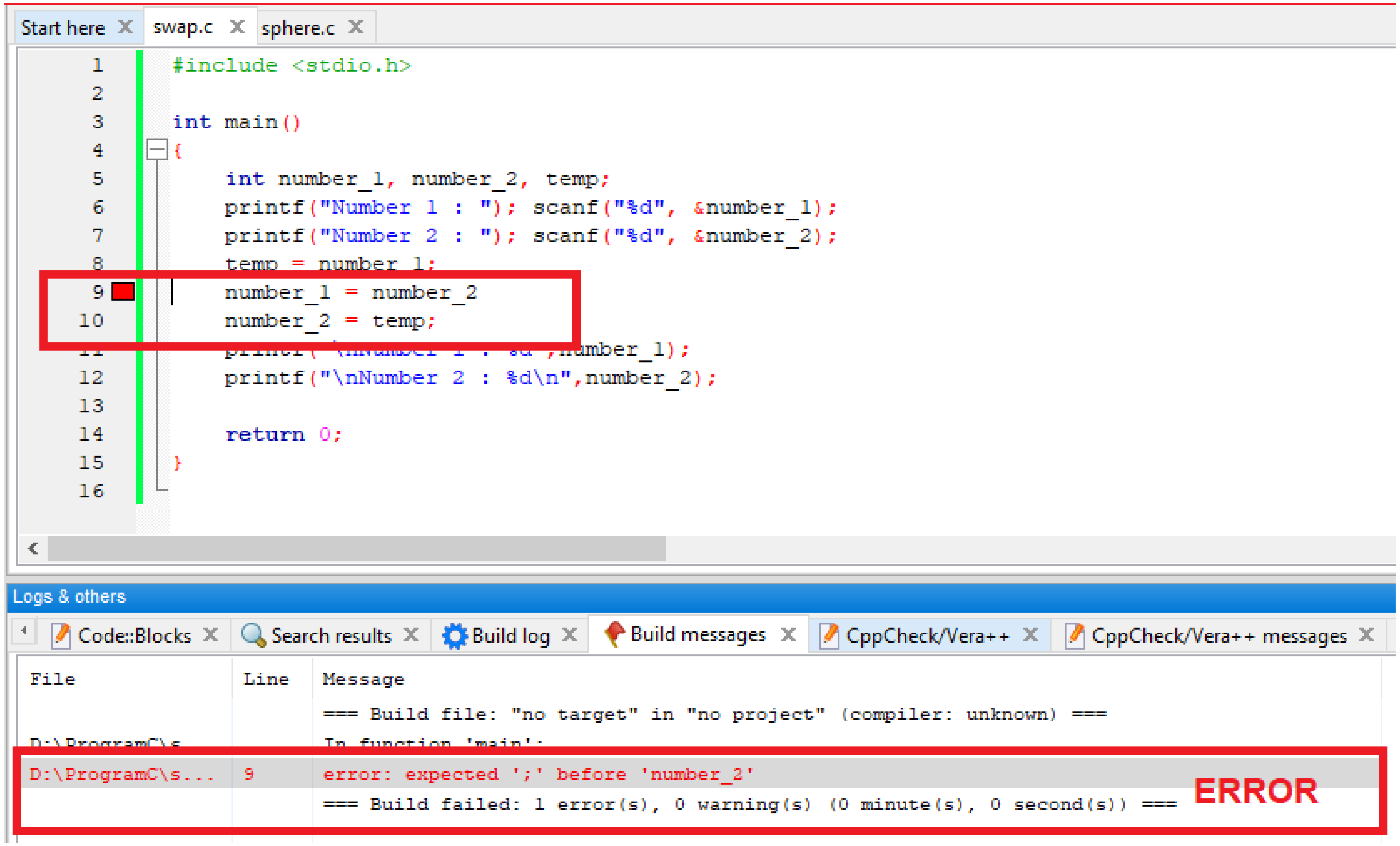
```
Radius : 15

Area : 2826.000000

Process returned 0 (0x0)   execution time : 78.980 s
Press any key to continue.
```

Compile and Run [F9]


*swap.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Build Ctrl-F9
Compile current file Ctrl-Shift-F9
Run Ctrl-F10
Build and run F9
Rebuild Ctrl-F11
Clean
Build workspace
Rebuild workspace
Clean workspace
Abort
Errors
Select target
Export Makefile
<global>
Management
Projects Files FSymbols Resou
View: All local symbols (workspa
Search:
*swap.c
sphere.c

```c
#include <stdio.h>

int main()
{
    int number_1, number_2, temp;
    printf("Number 1 : "); scanf("%d", &number_1);
    printf("Number 2 : "); scanf("%d", &number_2);
    temp = number_1;
    number_1 = number_2
    number_2 = temp;
    printf("\nNumber 1 : %d",number_1);
    printf("\nNumber 2 : %d\n",number_2);

    return 0;
}
```

Compile and Run [F9]

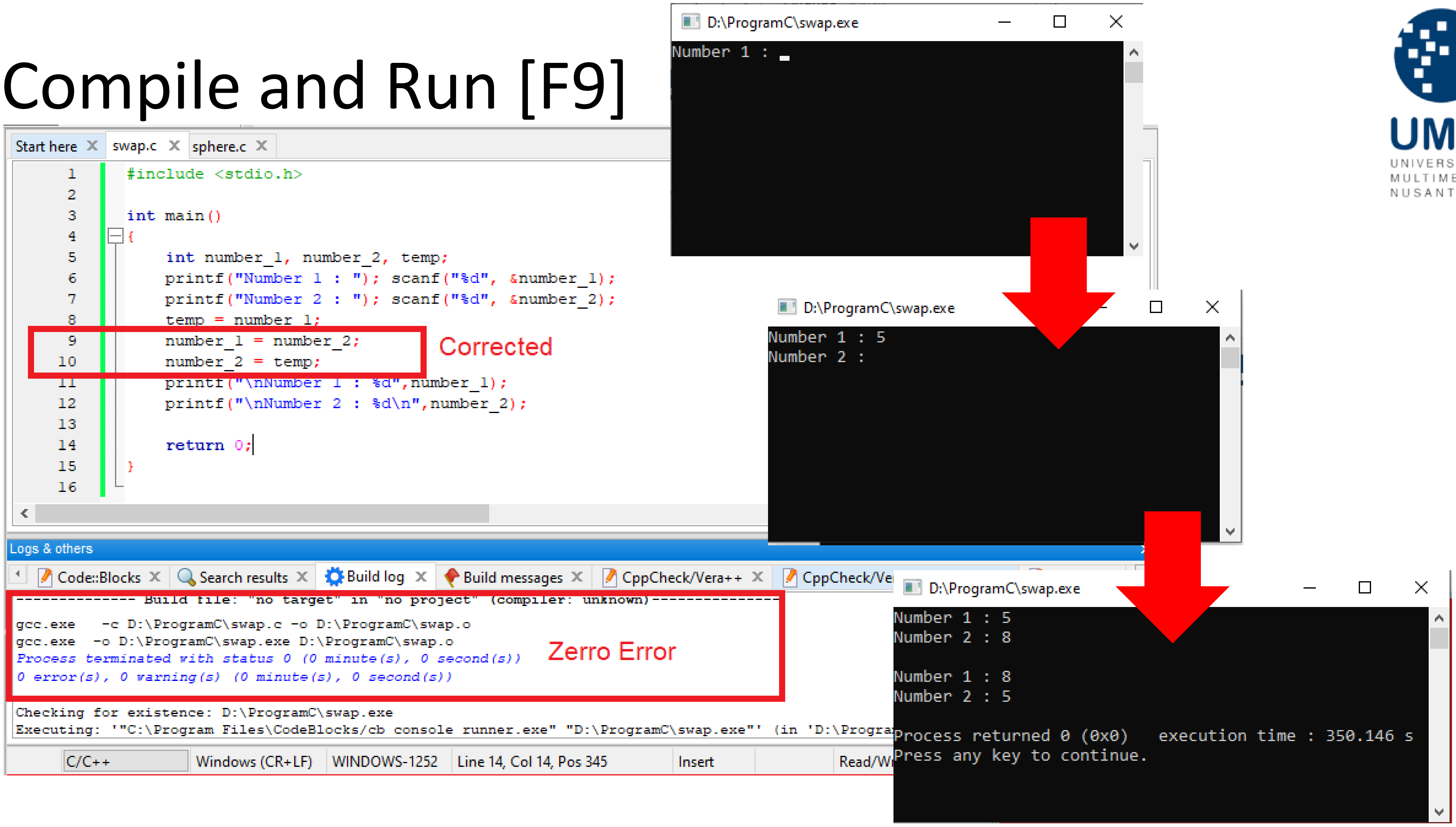


```
 1  #include <stdio.h>
 2
 3  int main()
 4  {
 5      int number_1, number_2, temp;
 6      printf("Number 1 : "); scanf("%d", &number_1);
 7      printf("Number 2 : "); scanf("%d", &number_2);
 8      temp = number_1;
 9      number_1 = number_2
10      number_2 = temp;
11      printf("\nNumber 1 : %d",number_1);
12      printf("\nNumber 2 : %d\n",number_2);
13
14      return 0;
15  }
16
```

```
File                Line   Message
                           === Build file: "no target" in "no project" (compiler: unknown) ===
D:\ProgramC\s...           In function 'main':
D:\ProgramC\s...    9      error: expected ';' before 'number_2'
                           === Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

ERROR

Compile and Run [F9]



```
Start here  swap.c  sphere.c

#include <stdio.h>

int main()
{
    int number_1, number_2, temp;
    printf("Number 1 : "); scanf("%d", &number_1);
    printf("Number 2 : "); scanf("%d", &number_2);
    temp = number_1;
    number_1 = number_2;
    number_2 = temp;
    printf("\nNumber 1 : %d",number_1);
    printf("\nNumber 2 : %d\n",number_2);

    return 0;
}
```

Corrected

Logs & others

```
-------------- Build file: "no target" in "no project" (compiler: unknown)---------------

gcc.exe   -c D:\ProgramC\swap.c -o D:\ProgramC\swap.o
gcc.exe  -o D:\ProgramC\swap.exe D:\ProgramC\swap.o
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))

Checking for existence: D:\ProgramC\swap.exe
Executing: '"C:\Program Files\CodeBlocks/cb_console_runner.exe" "D:\ProgramC\swap.exe"' (in 'D:\Progra
```

Zerro Error

C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 14, Col 14, Pos 345 | Insert | Read/W

D:\ProgramC\swap.exe

```
Number 1 : _
```

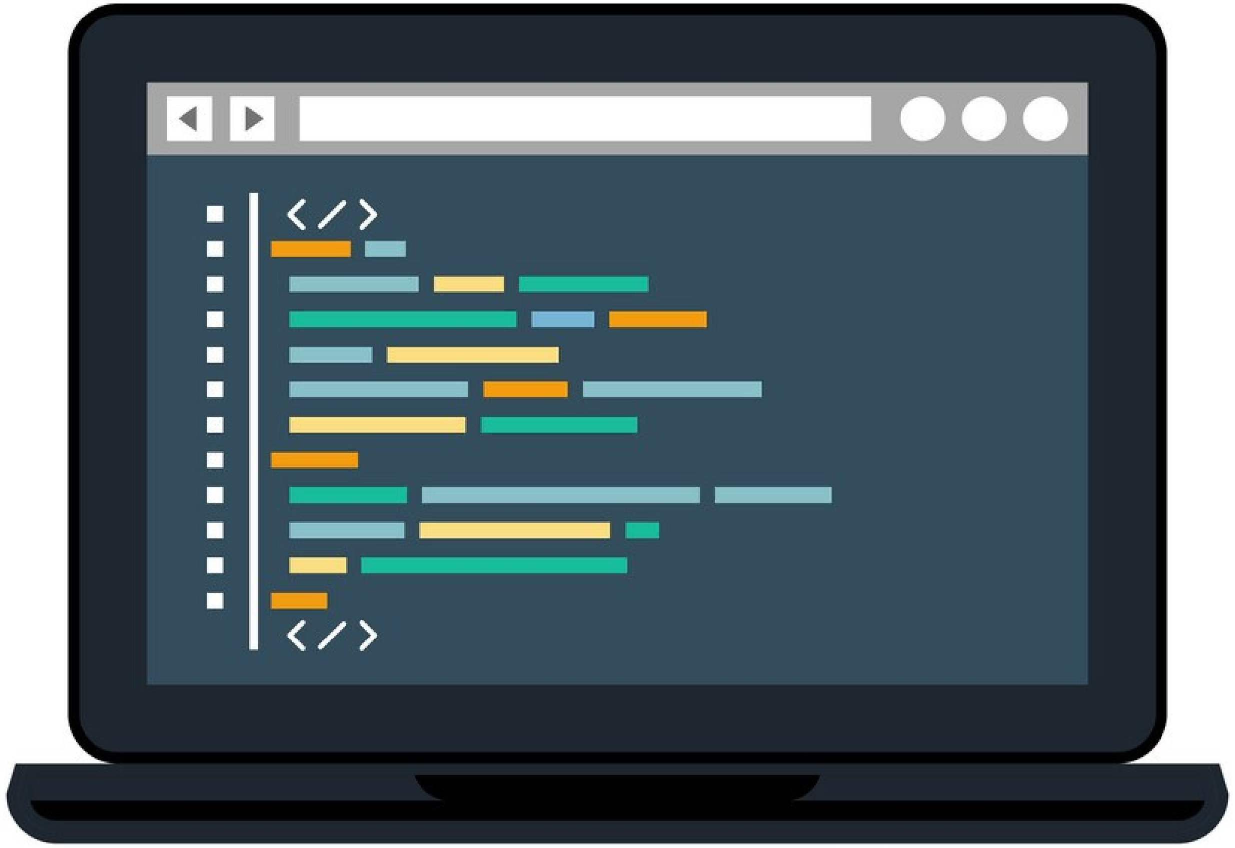
D:\ProgramC\swap.exe

```
Number 1 : 5
Number 2 :
```

D:\ProgramC\swap.exe

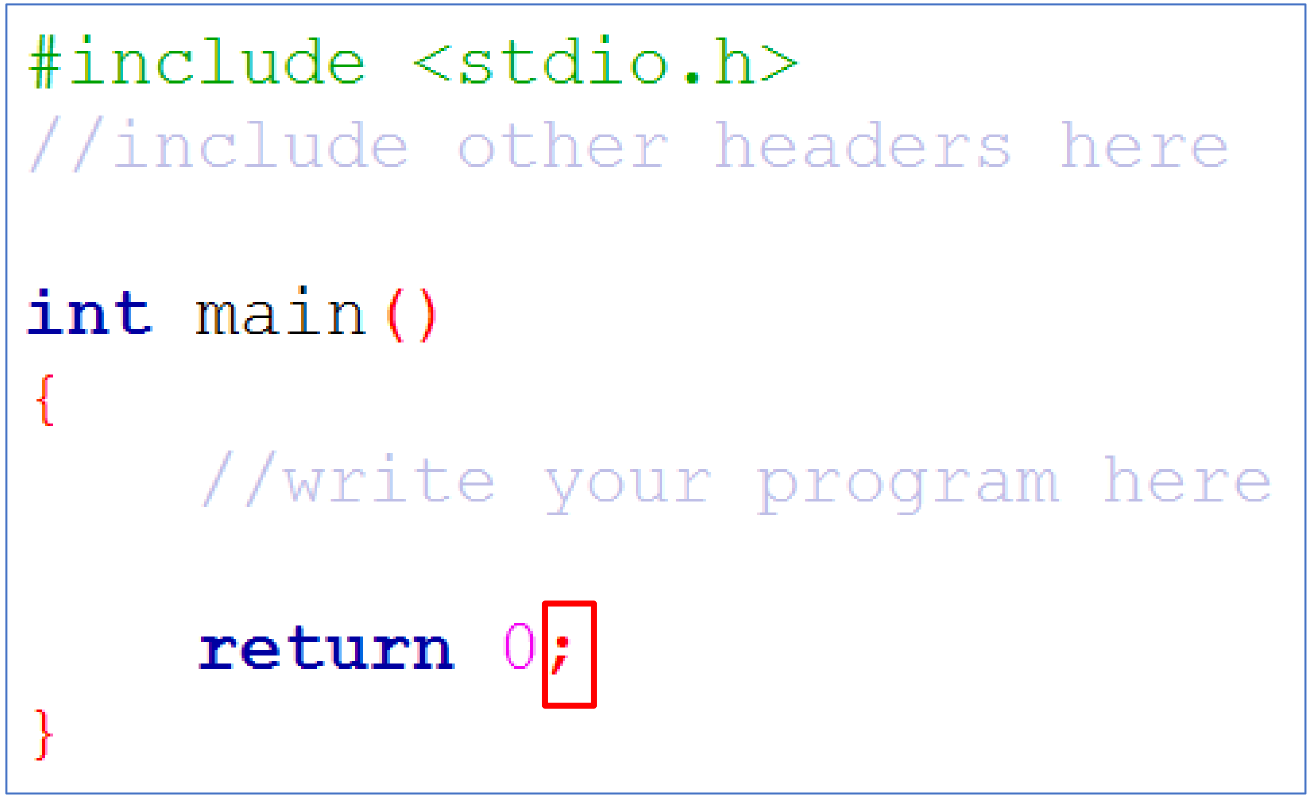
```
Number 1 : 5
Number 2 : 8

Number 1 : 8
Number 2 : 5

Process returned 0 (0x0)   execution time : 350.146 s
Press any key to continue.
```

CODING



Structure of C Program (Basic)

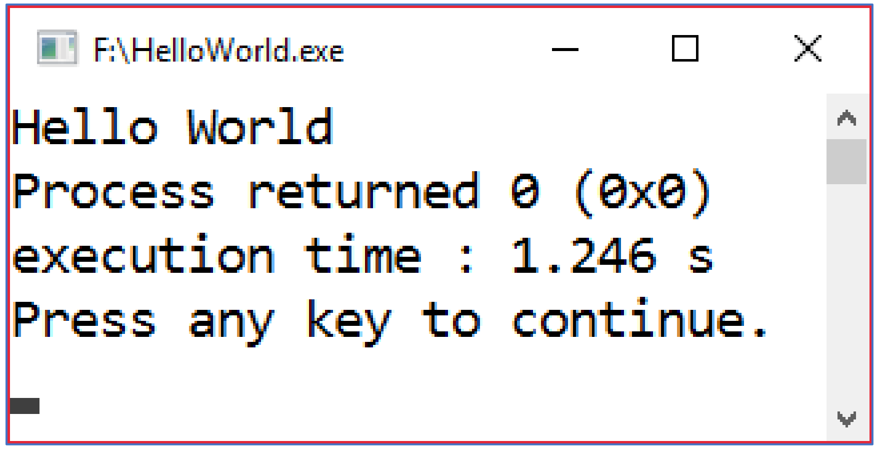
- Every statement **must end** with a **semicolon** (;)



```c
#include <stdio.h>
//include other headers here

int main()
{
    //write your program here

    return 0;
}
```

Hello World

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```



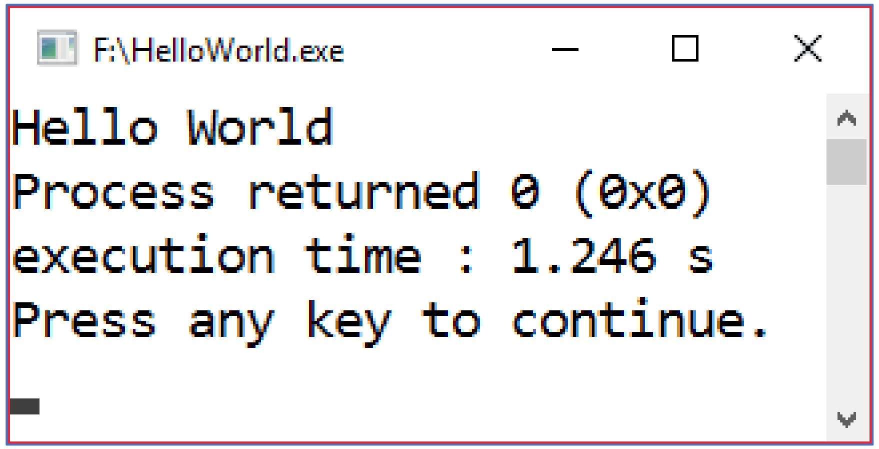
```
Hello World
Process returned 0 (0x0)
execution time : 1.246 s
Press any key to continue.
```

```
#include <stdio.h>
```

- Lines beginning with # are processed by the preprocessor before the program is compiled
- Tells the preprocessor to include the contents of the standard input / output header (<stdio.h>)
- <stdio.h> contains information used by the compiler when compiling calls to standard input / output library functions such as printf

Hello World

```c
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```



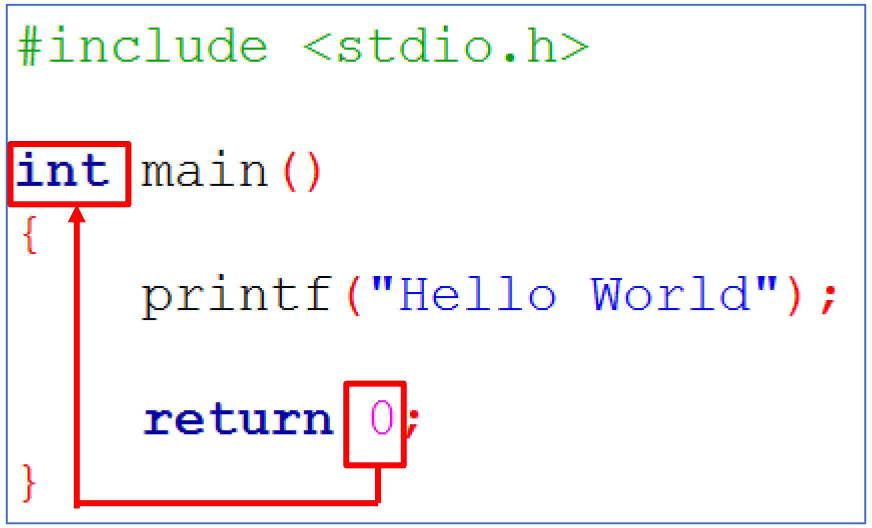
F:\HelloWorld.exe

```
Hello World
Process returned 0 (0x0)
execution time : 1.246 s
Press any key to continue.
```

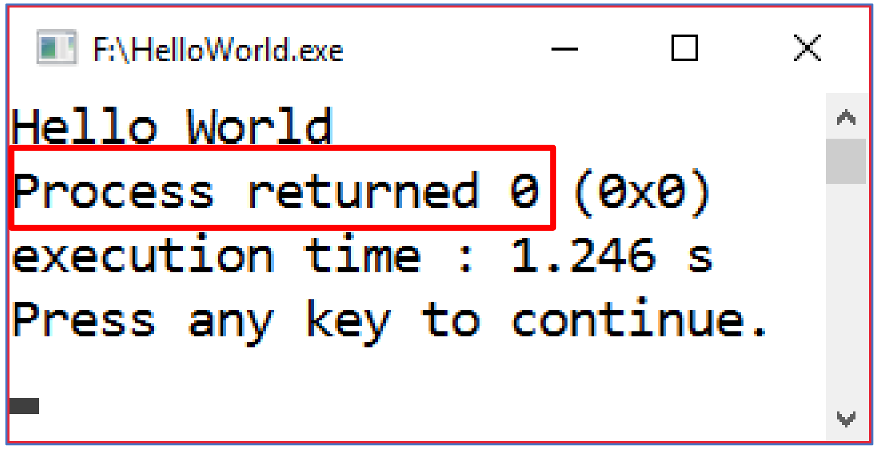
```
int main()
```

- A part of every C program
- The parentheses after main indicate that main is a program building block called a function
- C programs contain one or more functions, **one of which must be main**
- Every program in C **begins executing** at the function **main**

Hello World



```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```



```
Hello World
Process returned 0 (0x0)
execution time : 1.246 s
Press any key to continue.
```

`int main()`

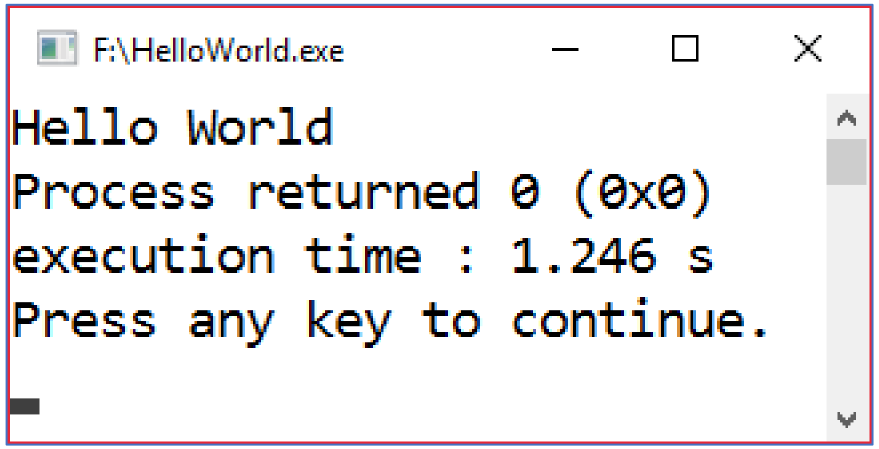
- The keyword int to the left of main indicates that main returns an integer (whole number) value

`return 0;`

- The keyword return is used to **exit a function**
- When the return statement is used at the end of main, the value 0 indicates that the program has terminated successfully

Hello World

```c
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```



```
F:\HelloWorld.exe
Hello World
Process returned 0 (0x0)
execution time : 1.246 s
Press any key to continue.
```

`{`

- A left brace begins the body of every function

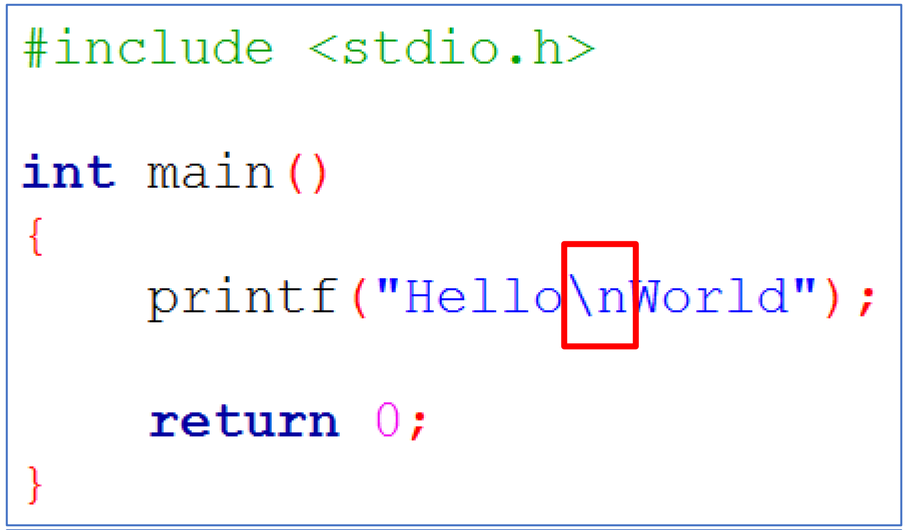
`}`

- A corresponding right brace ends each function

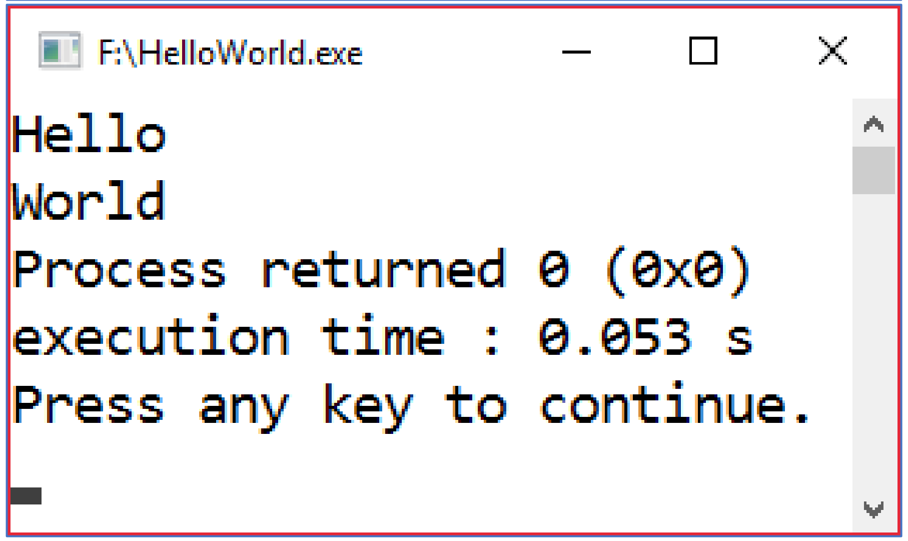
`printf(“Hello World”);`

- Instructs the computer to print on the screen the string of characters marked by the quotation marks

Escape Sequence



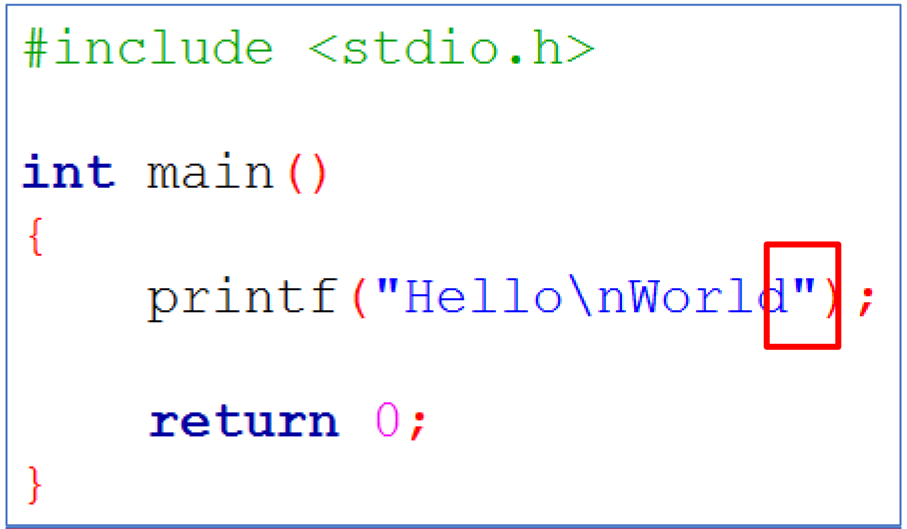
```
#include <stdio.h>

int main()
{
    printf("Hello\nWorld");

    return 0;
}
```



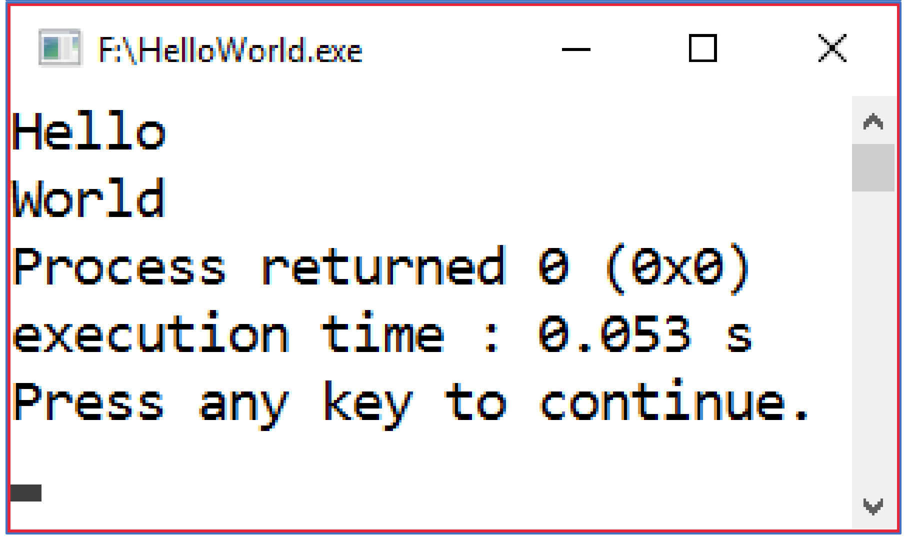
```
F:\HelloWorld.exe
Hello
World
Process returned 0 (0x0)
execution time : 0.053 s
Press any key to continue.
```

- Notice that the characters **\n** were not printed on the screen
- The backslash (\ ) is called an **escape character**
- When encountering a backslash in a string, the compiler looks ahead at the next character and combines it with the backslash to form an **escape sequence**

Escape sequence



```c
#include <stdio.h>

int main()
{
    printf("Hello\nWorld");

    return 0;
}
```



F:\HelloWorld.exe

```
Hello
World
Process returned 0 (0x0)
execution time : 0.053 s
Press any key to continue.
```

Escape Sequence	Description
\n	**Newline** Position the cursor at the beginning of the next line
\t	**Horizontal Tab** Move the cursor to the next tab stop
\a	**Alert** Sound the system bell
\\	**Backslash** Insert a backslash character in a string
\”	**Double Quote** Insert a double-quote character in a string

Source Code Formatting

- **Conventions**
  - Start **a new line** for each new declaration and statement
  - Use **indentation** to reflect the nested structure of block statements

Program documentation (code commenting)

- Insert comments to document programs and improve program readability
- Comments do not cause the computer to perform any action when the program is run
- Comments are ignored by the C compiler and do not cause any machine-language object code to be generated

```
//one line comment

/*
     multiple
     line
     comments
*/
```

```
#include <stdio.h>

int main()
{
    //variable declaration
    float radius, area;

    //input: get the radius from user
    printf("Radius: ");
    scanf("%f",&radius);

    /*process:
      calculate the area of a sphere*/
    area = 4 * 3.14 * radius * radius;

    //output: print the area
    printf("\nArea: %f\n",area);

    return 0; //indicate that program ended successfully
}
```

Identifier

- A series of characters consisting only of **letters**, **digits**, and **underscores** that **does not begin with a digit**
- Can be of any length, but only the first 31 characters are required to be recognized by C compilers according to the C standard
- **Case sensitive**: uppercase and lowercase letters are different

$$\boxed{A \neq a}$$

Keywords (reserved words)

- Have special meaning to the C compiler
- Be careful not to use these as identifiers such as variable names

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

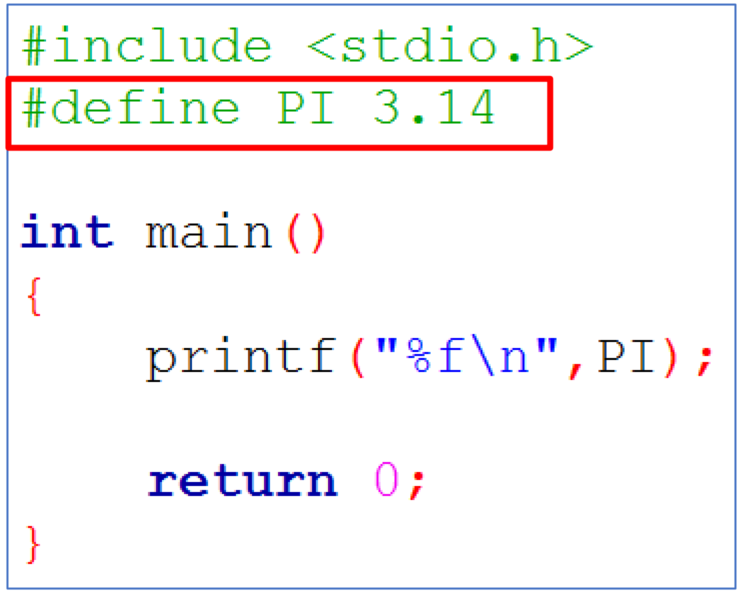
Data types

Data Type	Keyword	Size (bit)
Character	char unsigned char	8 8
Short Integer	short unsigned short	16 16
Integer	int unsigned int	16 / 32 16
Long Integer	long unsigned long	32 32
Long Long Integer	long long unsigned long long	64 64
Single-precision floating point	float	32
Double-precision floating point	double	64
Extended-precision floating point	long double	80 / 96

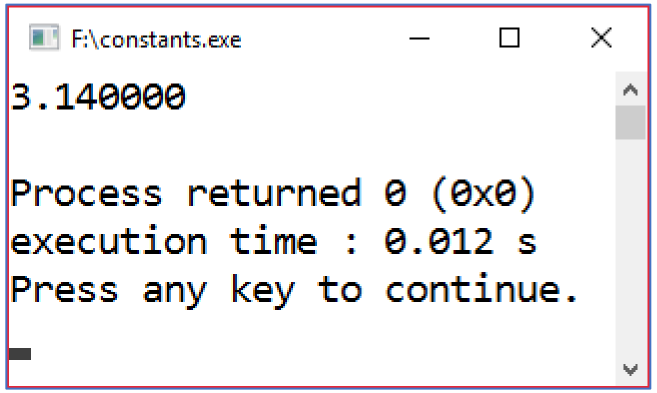
Constants

- Preprocessor Directives

```
#define PI 3.14
```



```
#include <stdio.h>
#define PI 3.14

int main()
{
    printf("%f\n",PI);

    return 0;
}
```



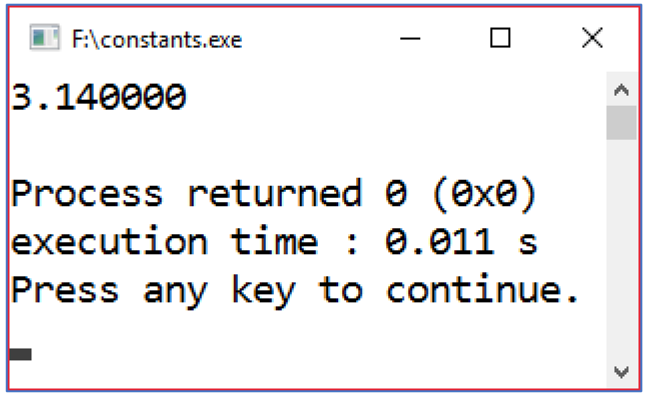
```
F:\constants.exe
3.140000

Process returned 0 (0x0)
execution time : 0.012 s
Press any key to continue.
```

Constants

- Constant Modifier

```
const float PI = 3.14;
```

```
#include <stdio.h>

int main()
{
    const float PI = 3.14;

    printf("%f\n",PI);

    return 0;
}
```



F:\constants.exe

```
3.140000

Process returned 0 (0x0)
execution time : 0.011 s
Press any key to continue.
```

Variable

- A location in memory where a value can be stored for use by a program
- Variable Declaration

```
datatype variable_name [= init_value] [, variable_name ...];
```

- Example

```
int i1 = 5, i2;
float f1, f2 = 3.8;
char c1 = 'D';
```

```
float f1;
float f2 = 3.8;
float f3 = 92.7;
```

REFERENCES

- Hanly, Jeri R. and Koffman, Elliot B., 2013, Problem Solving and Program Design in C, Seventh Edition, Pearson Education, Inc.
- Deitel, Paul and Deitel, Harvey, 2016, C How to Program, Eighth Edition, Pearson Education, Inc.

NEXT WEEK’S OUTLINE

1. Assigment operator
2. Identifiers and Keywords
3. Operators and Operations
4. Memory Concepts
5. Function Prototype
6. Formatted & Unformatted Input
7. Formatted & Unformatted Output

Visi

Menjadi Program Studi Strata Satu Informatika **unggulan** yang menghasilkan lulusan **berwawasan internasional** yang **kompeten** di bidang Ilmu Komputer (*Computer Science*), **berjiwa wirausaha** dan **berbudi pekerti luhur**.

INFORMATIKA
UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

Misi

1. Menyelenggarakan pembelajaran dengan teknologi dan kurikulum terbaik serta didukung tenaga pengajar profesional.
2. Melaksanakan kegiatan penelitian di bidang Informatika untuk memajukan ilmu dan teknologi Informatika.
3. Melaksanakan kegiatan pengabdian kepada masyarakat berbasis ilmu dan teknologi Informatika dalam rangka mengamalkan ilmu dan teknologi Informatika.