A Review of C Programming and Design

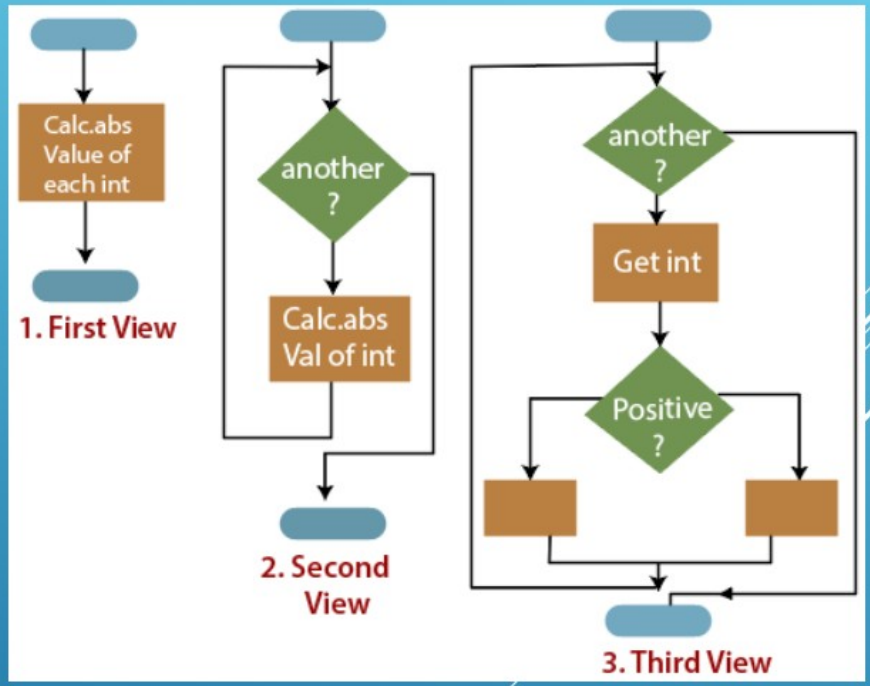
Professor: Yu-Chien Ko

Teaching-Learning Material

- Moodle of Chung Hua University
- GitHub of Teaching-Learning Material
- C Examples of Programiz
-

STRUCTURED PROGRAMMING

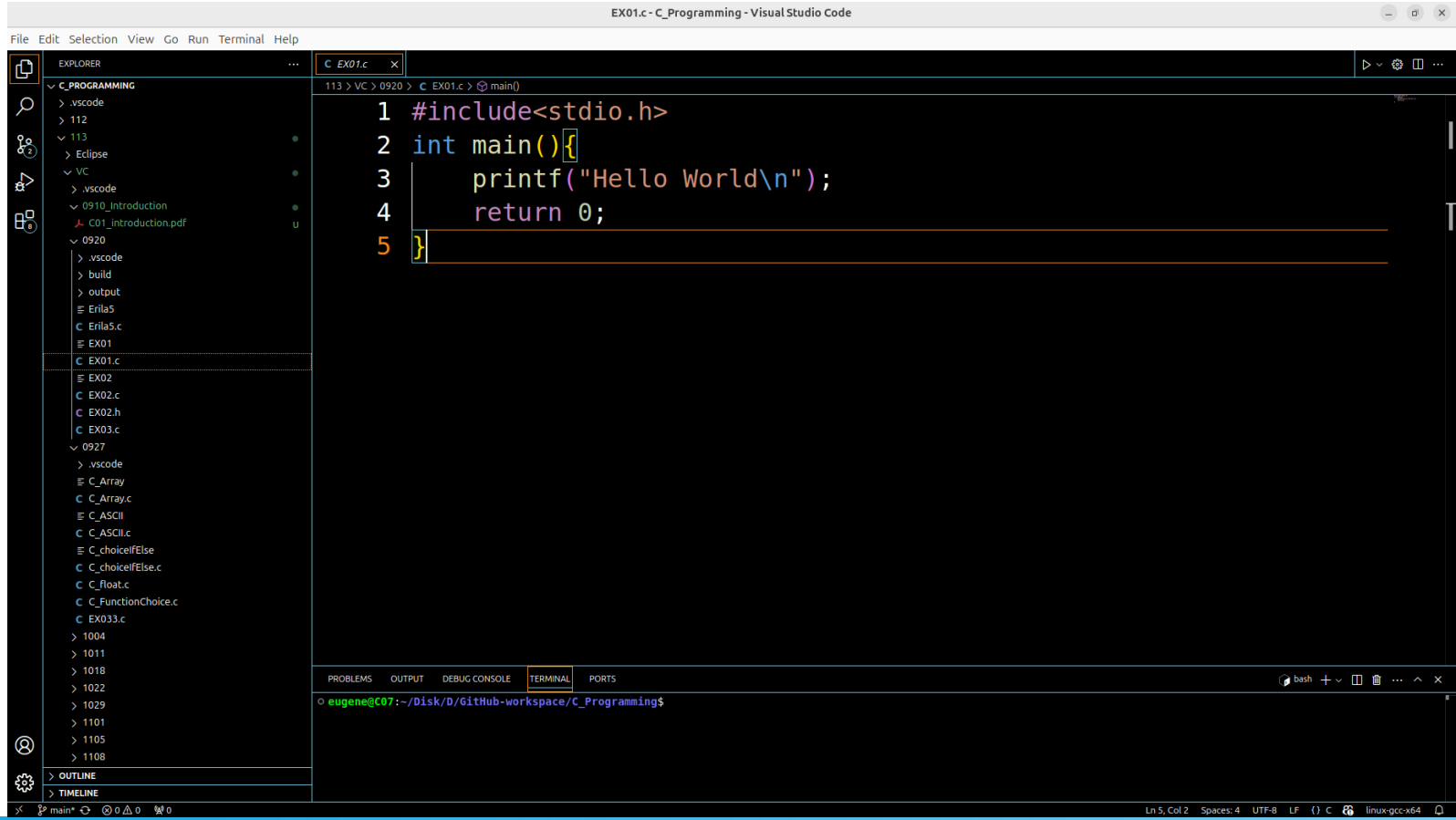
- A programming paradigm aims at improving the clarity, quality, and development time of a computer program. It emphasizes the use of fundamental structures such as selection (if/then/else), Iteration (while and for), Sequence, and subroutines. This approach helps in organizing code in a way that is easier to understand, maintain, and debug



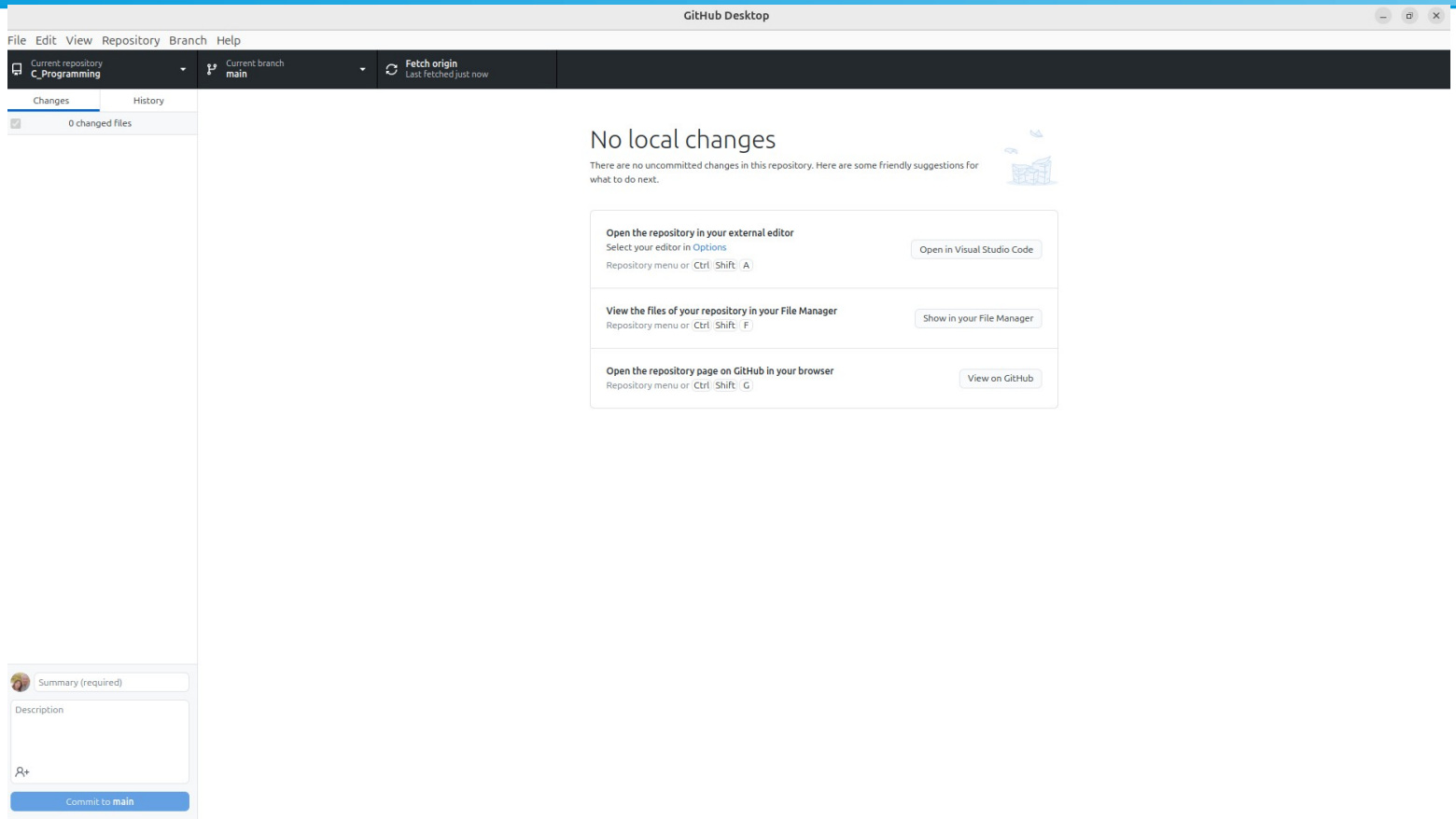
https://www.youtube.com/watch?v=TmtyFcLWXwo&t=12s&ab_channel=EzEdChannel

Visual Code

- Programming Tool of Microsoft
- Link from GitHub Desktop



GitHub Desktop



Block Structured Programming

- Block Example 1
- Block Example 2
- Block Example 3

Selection Structured Programming

- GitHub Example

Logical Programming

- Example 1
- Example 2
- Example 3
-
-

Loop Programming

- Example 1
- Example 2
- Example 3

C Project

- GNU Makefile: It tells make how to compile and link a program. It defines a set of rules and dependencies that specify how to compile source code, link object files, and generate executable files or libraries.
- GNU make tool: An utility automatically building C projects such as helping developers compile and build software projects efficiently.
- Example

Makefile EX_1

```
# https://www.youtube.com/watch?v=CRlqU9XzVr4&ab_channel=JacobSorber

BIN = hellomake

SOURCES = hellomake.c hellofunc.c

OBJECTS = $(SOURCES:.c=.o)

CFLAGS = -g3 -Wall # debugging information # telling the compiler to enable a standard set of warnings about potential issues

ALL: $(BIN)

$(BIN):$(OBJECTS)

	gcc $(CFLAGS) $(OBJECTS) -o $(BIN)

%.o: %.c

	gcc $(CFLAGS) -c $< -o $@

clean: rm -f $(BIN) $(OBJECTS)
```

Makefile EX_2

```
BIN = bin/hellomake

SOURCES = src/hellomake.c src/hellofunc.c

OBJECTS = obj/hellomake.o obj/hellofunc.o

CFLAGS = -g3 -Wall

ALL: $(BIN)

$(BIN):$(OBJECTS)

	gcc $(CFLAGS) $(OBJECTS) -o $(BIN)

obj/%.o: src/%.c

	gcc $(CFLAGS) -c $< -o $@

clean: rm -f $(BIN) $(OBJECTS)
```

Makefile EX_3

```
# Makefile

# https://www.youtube.com/watch?v=CRlqU9XzVr4&ab_channel=JacobSorber

BINDIR=bin

BIN = $(BINDIR)/hellomake

SRCDIR=src

OBJDIR=obj

SOURCES = $(wildcard $(SRCDIR)/*.c)

OBJECTS = $(patsubst $(SRCDIR)/%.c, $(OBJDIR)/%.o, $(SOURCES))

CFLAGS = -g3 -Wall

ALL: $(BIN)

$(BIN):$(OBJECTS)

        gcc $(CFLAGS) $(OBJECTS) -o $(BIN)

$(OBJDIR)/%.o: $(SRCDIR)/%.c

        gcc $(CFLAGS) -c $< -o $@

clean: rm -f $(BINDIR)/* $(OBJDIR)/*
```

A simple makefile

A simple makefile consists of “rules” with the following shape:

```
target … : prerequisites …
        recipe
        …
        …
```

- A target is usually the name of a file that is generated by a program; examples of targets are executable or object files.
- A prerequisite is a file that is used as input to create the target.
- A recipe is an action that make carries out.