IPC144
Introduction to C Programming

Week-4

Logic:

*** Selection**

*** Iteration**

Logic…

Preparation

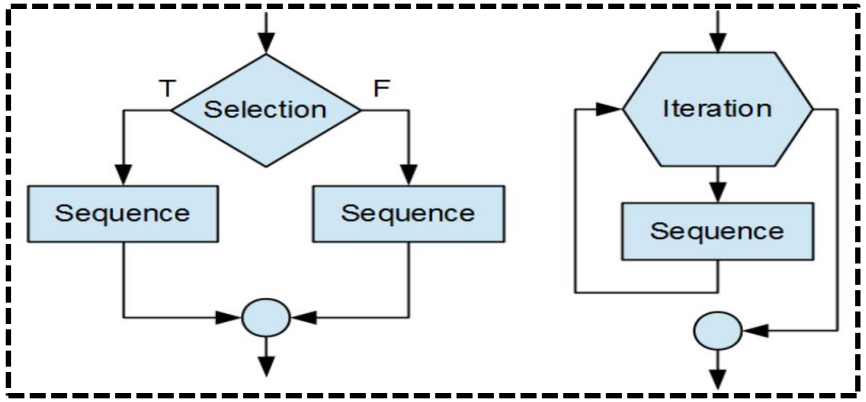
BEFORE writing code use "**Pseudo-Code**"

- Outline the processes required
- Define the sequence of instructions (as they should occur)
- Review instructions and modify as necessary

```
1) declare variables for quarters and nickels
2) calculate the number of quarters in the change
3) calculate the remainder to be returned in nickels
4) output the change in quarters and nickels
```

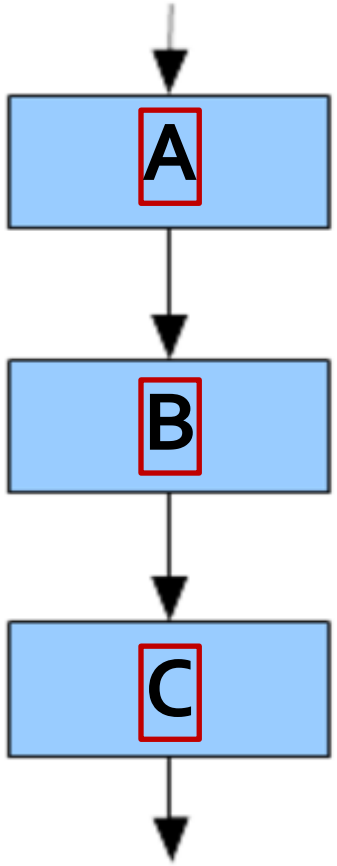
Optionally consider a **flowchart**

- Flowcharts aren't as quick to work with if major changes are required, but are really good at graphically presenting the flow



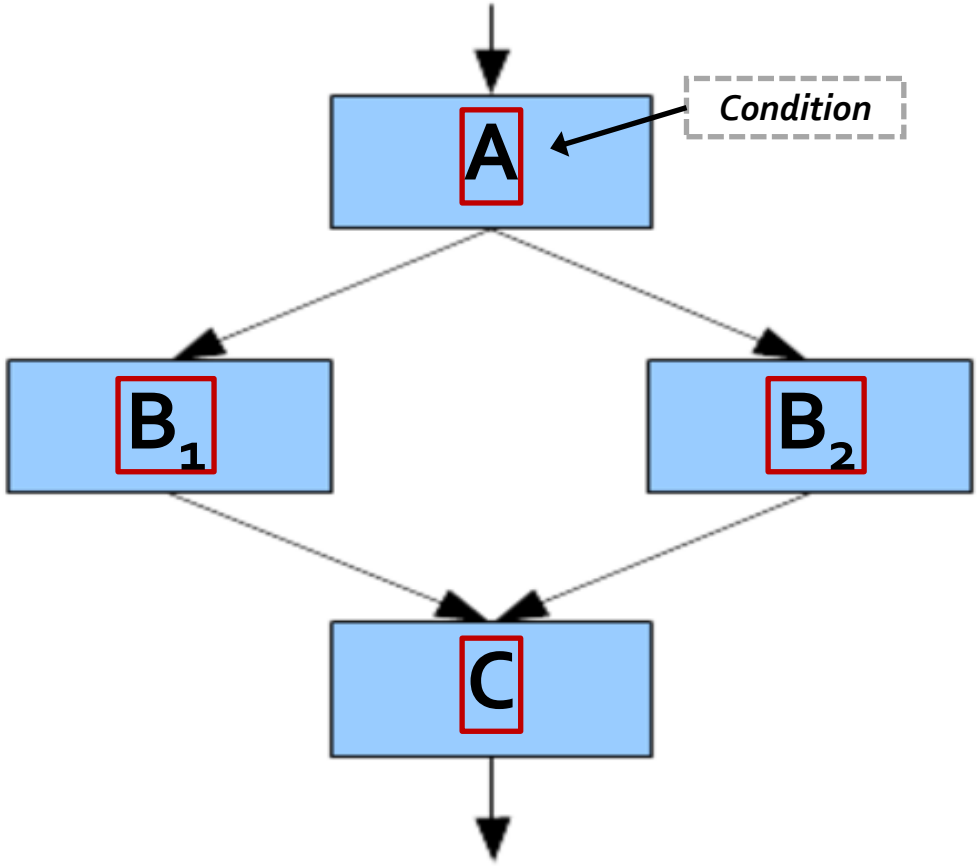
Logic… (Structured Programming)

Sequential



Sequence

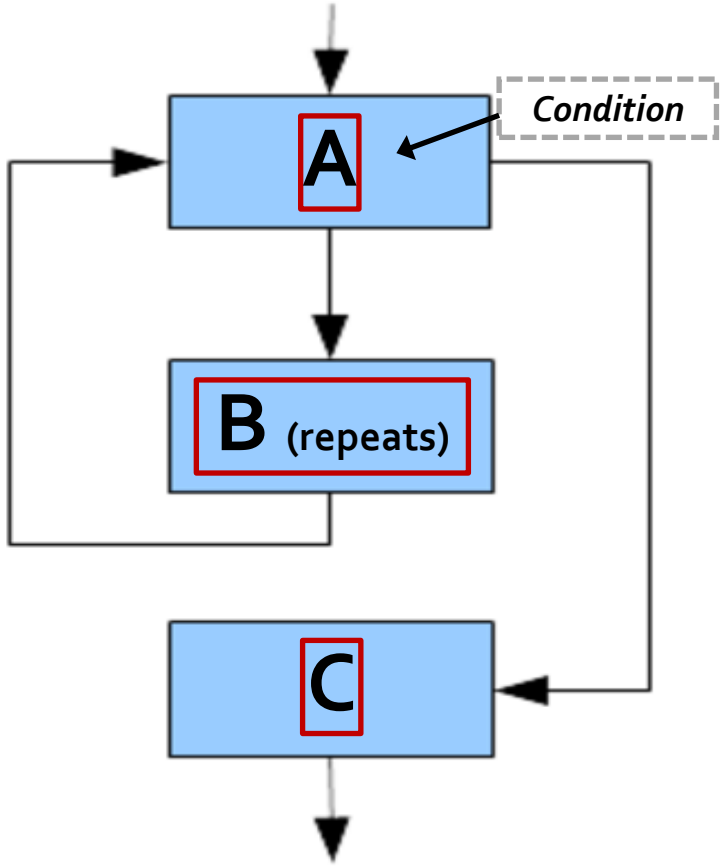
Conditional Path(s)



Example **if / else if / else & switch**

Selection

Conditional Repeating



Example **while / do while / for**

Iteration

Logic…

Optional Path (singular **if** statement)

Display an optional message based on a condition
*(**Note**: there is no alternate message when the condition is not met)*

Scenario-1: Display "Good job!" only when a grade is >= 80%

```
if( percent >= 80 )
{
      printf("Good Job");
}
```

Scenario-2: Display "Good job!" and on the next line display "Keep up the good work!" only when a grade is >= 80%

```
if( percent >= 80 )
{
      printf("Good Job!\n");
      printf("Keep up the good work!");
}
```

Logic…

Alternative Path (**if / else** statement)

Display a mandatory custom message based on a condition
*(**Note**: a message is displayed regardless of the conditional outcome)*

Scenario: Display "Good job!" when a grade is >= 80% otherwise "Satisfactory" (when grade is < 80%)

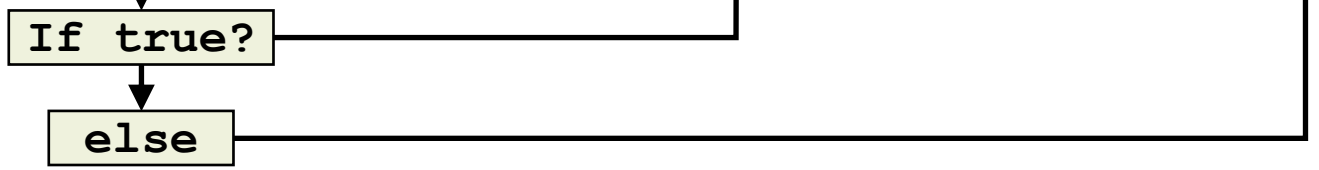
```
if( percent >= 80 )
{
      printf("Good Job");
}
else
{
      printf("Satisfactory");
}
```

Logic…

Conditional/Ternary Expression (inline **if / else** alternative)

To be used carefully and sparingly as it can be difficult to read

```
condition ? operand : operand
```



Scenario: Display 12-hour time format based on a 24 hour value and show am/pm accordingly

```
int hours = 14, mins = 15;
char amPm = hours < 12 ? 'A' : 'P';
printf("Time:%2d:%2d %cM.\n", hours < 12 ? hours : hours - 12, mins, amPm);
```

Process Flow:
- If the hours value is less than 12 then assign letter 'A'
- Otherwise assign letter 'P'

Process Flow:
- If the hours value is less than 12 then use the hours variable without changes
- Otherwise subtract 12 from the hours variable

Logic…

Alternative Path (**if / else if / else** statement)

Multiple conditions

Scenario: Display "Poor job!" when a grade is < 50%
Display "Satisfactory job!" when a grade is >=50% but <80%
Display "Good job!" when grade is >=80 but < 90%
Display "Excellent job!" when grade is >=90

```
if( percent < 50 )
{
        printf("Poor job!");
}
else if ( percent < 80 )
{
        printf("Satisfactory job!");
}
else if ( percent < 90 )
{
        printf("Good job!");
}
else
{
        printf("Excellent job!");
}
```

Logic…

Alternative Path (switch…case…default)

*Multiple conditions (matching **CONSTANT** value)* ***!!! INTEGRAL ONLY !!!***

Scenario (pseudo code):

Display "Poor job!" when grade letter is 'F'
Display "Satisfactory job!" when grade letter is 'D' or 'C'
Display "Good job!" when grade letter is 'B'
Display "Excellent job!" when grade letter is 'A'
Display "Not graded" when grade is anything else

```
switch(grade)
{
case 'F':
       printf("Poor job!");
       break;
case 'D':
case 'C':
       printf("Satisfactory job!");
       break;
case 'B':
       printf("Good job!");
       break;
case 'A':
       printf("Excellent job!");
       break;
default:
       printf("Not graded");
}
```

Logic…

do while (iteration/loop)

Do what's in a block of code **at least once** and then **repeat** as long as the **condition(s) are true**

Scenario (pseudo code):

1. Display a greeting
2. Display a prompt for the user to enter a whole number
3. Get the input from the user
4. Accumulate (sum) the value(s) entered into a variable
5. Display a prompt asking the user if another entry is desired (1=yes | 0=no)
6. Get the input from the user
7. If the input is **1** then repeat number entry (#2)
8. If the input is **0** then Display the sum entered and exit

```
int total = 0, entry, flag;

printf("Number Accumulator\n");
printf("------------------\n\n");

do
{
  printf("Enter a whole number:");
  scanf("%d", &entry);

  total += entry;

  printf("Enter another number (1=yes|0=no)?:");
  scanf("%d", &flag);

} while( flag == 1 );

printf("The total sum entered is:%d\n\n", total);
```

Logic…

while (iteration/loop)

Repeat what's in a block of code **only while the condition(s) are true**

Scenario (pseudo code):

1. Display a greeting
2. Display a prompt for the user to enter a whole number
3. Get the input from the user
4. Accumulate (sum) the value(s) entered into a variable
5. Display a prompt asking the user if another entry is desired (1=yes | 0=no)
6. Get the input from the user
7. If the input is **1** then repeat number entry (#2)
8. If the input is **0** then Display the sum entered and exit

```
int total = 0, entry;
int flag = 1;                    // initialize to true!

printf("Number Accumulator\n");
printf("-------------------\n\n");

// To ensure it runs once; condition must be true on first run
while( flag == 1 )
{
  printf("Enter a whole number:");
  scanf("%d", &entry);

  total += entry;

  printf("Enter another number (1=yes|0=no)?:");
  scanf("%d", &flag);
}

printf("The total sum entered is:%d\n\n", total);
```

Logic…

for (iteration/loop)

Initialization performed **ONCE** at the start of the for loop cycle

```
for (initialization; condition; change)
```

Condition is **evaluated at the BEGINNING of each loop cycle**
If the condition evaluates to true (1) execute the code within the for block
If the condition evaluates to false (0) exit the for block and continue with the program

change is **evaluated at the END of each loop cycle**
Best used when directly related to the condition clearly showing how each cycle will advance

Logic…

for

Repeat what's in a block of code **only while the condition(s) are true**

Scenario (pseudo code):

1. Display a greeting
2. Get 5 numbers from the user to accumulate
3. Display a prompt for the user to enter a whole number
4. Get the input from the user
5. Accumulate (sum) the value entered into a variable
6. After the 5th entry display the sum entered and exit

```
int total = 0, entry;
int i;              // iterator for the loop
int max = 5;        // number of iterations

printf("Number Accumulator\n");
printf("------------------\n\n");

for( i=0; i < max; i++ )
{

  printf("Enter a whole number:");
  scanf("%d", &entry);

  total += entry;
}

printf("The total sum entered is:%d\n\n", total);
```

Logic…

for — **Repeat** what's in a block of code **only while the condition(s) are true**

Order or execution:

1. *For initialization section*: `i` get initialized to `0`
2. *For condition section*: **i < max**
   - If true (evaluates to `1`): see step #3
   - If false (evaluates to `0`): see step #6
3. Execute code block:
   - Prompt user for a whole number
   - Get the input from the user
   - Add input value to total accumulator variable
4. *For change section*: `i++` (increment `i` by one:`i=i+1`)
5. See step #2
6. Exit the for block and continue with the program

```
        1      2      5      4
for( i=0; i < max; i++ )
{
3   printf("Enter a whole number:");
    scanf("%d", &entry);

    total += entry;
}

6 // The rest of the program…
```