PROGRAMMING

CT103

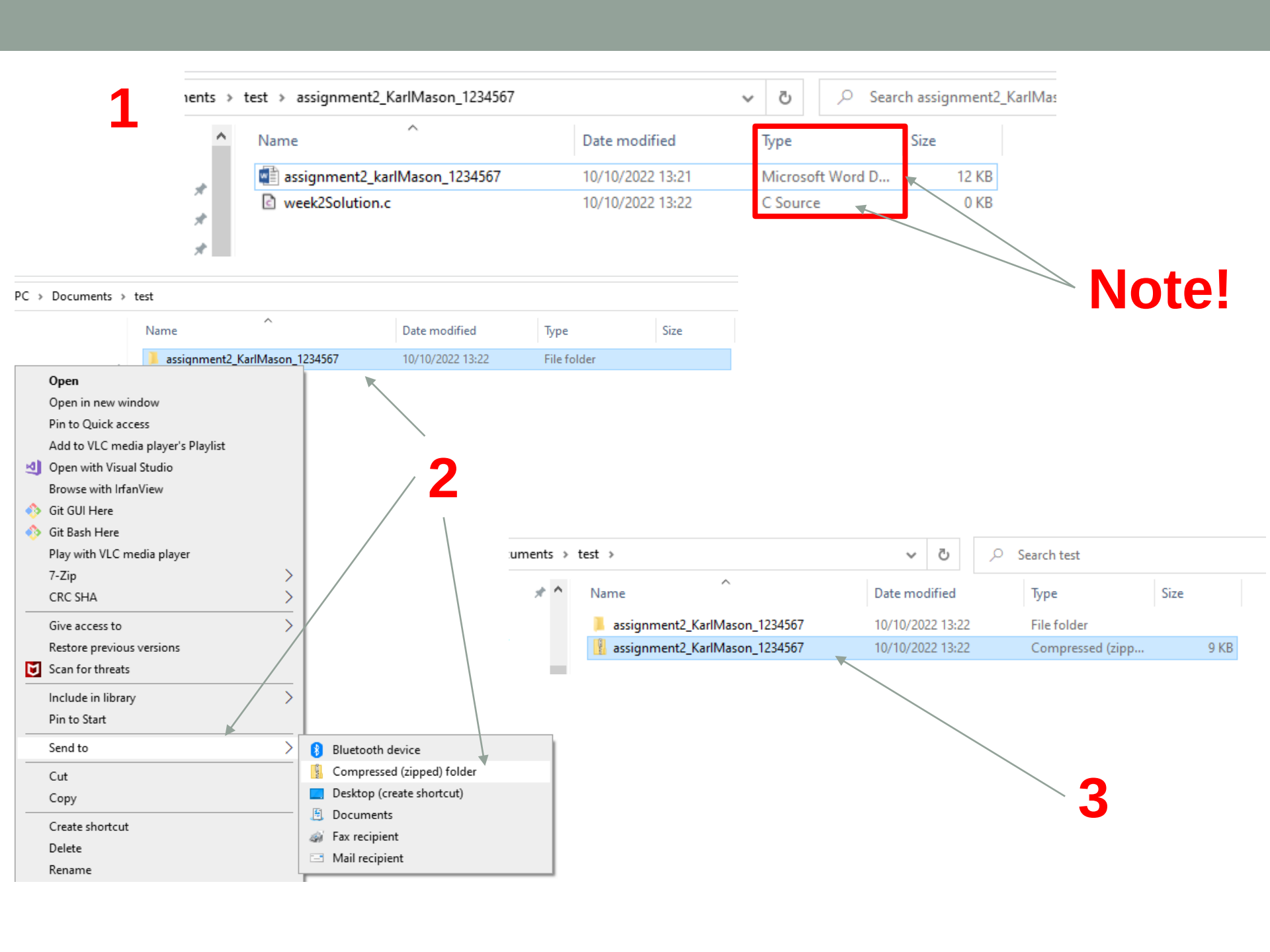
Week 3b

Note on Lab Assignments

- Please make sure to bring your laptop if you can.
- Submit your **.c file** and **.doc file** in a **zipped** up **folder**.
- If you receive an email from me or the lab instructors, requesting the correct files. Please **respond as soon as possible** to ensure the correct files are marked.
- Make sure your code in your .c file matches your code in your .doc file.

Note on Lab Assignments

- Make sure you do the assignments yourself. It is fine to ask each other questions or use C code I provide in lectures.
- Do not copy another students assignment. Plagiarism is taken seriously by the university.
- If instances of plagiarism are detected, all students involved will receive a grade of zero for the assignment and may be subject to further disciplinary proceedings.


1
ents › test › assignment2_KarlMason_1234567
Search assignment2_KarlMas
Name
Date modified
Type
Size
assignment2_karlMason_1234567
10/10/2022 13:21
Microsoft Word D...
12 KB
week2Solution.c
10/10/2022 13:22
C Source
0 KB
Note!
PC › Documents › test
Name
Date modified
Type
Size
assignment2_KarlMason_1234567
10/10/2022 13:22
File folder
Open
Open in new window
Pin to Quick access
Add to VLC media player's Playlist
Open with Visual Studio
Browse with IrfanView
Git GUI Here
Git Bash Here
Play with VLC media player
7-Zip
CRC SHA
Give access to
Restore previous versions
Scan for threats
Include in library
Pin to Start
Send to
Cut
Copy
Create shortcut
Delete
Rename
Bluetooth device
Compressed (zipped) folder
Desktop (create shortcut)
Documents
Fax recipient
Mail recipient
2
uments › test
Search test
Name
Date modified
Type
Size
assignment2_KarlMason_1234567
10/10/2022 13:22
File folder
assignment2_KarlMason_1234567
10/10/2022 13:22
Compressed (zipp...
9 KB
3

Lecture Content

- Last lecture (Week 3a):
  - Boolean Logic.
  - Switch Statements.
  - Characters.
- Today's evening lecture (Week 3b):
  - Loops.
  - While loops.
  - Do while loops.
  - Example C program.

WHY DO WE NEED LOOPS?

Loops

- Up until now we have not actually looked at any C programs that use loops.
- Loops are useful if we want to do the same task more than once.
- If we did not have loops, we would have to rewrite the same code over and over.
  - This would be time consuming, unreadable and difficult to change.

Motivating Loops Example

- How would I write a program that would do the following 3 times **without using loops**.
  - Read in two numbers.
  - Add them together.
  - Print the result.

Motivating Loops Example

- You could do the following:

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);
}
```

Motivating Loops Example

- When you run the code, it works and gives the following output:

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);
}
```

```
Enter number 1:5
Enter number 2:3
The sum is 8
Enter number 1:4
Enter number 2:1
The sum is 5
Enter number 1:6
Enter number 2:8
The sum is 14
```

Motivating Loops Example

- There are plenty of problems with this:
- The code is longer than it needs to be.
- What if we want to change it so that we are subtracting numbers instead of adding numbers?
- This is doable for repeating this task 3 times, what if we want to do it 100 times? Or 10000 times?

- Loops are a way of solving these issues!

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);
}
```

WHILE LOOPS

While Loops

- The first type of loop we will cover is called the **while loop**.

- The while loop will repeat a block of code over and over while some condition is true.

While Loops Template

- While loops have the following structure:

```
while (condition) {
    // Do something
}
```

- We have some **condition**, e.g. number<10.
- While this condition is **True**, whatever is inside the curly brackets {} gets executed.
- This is useful for doing something more than once!

While Loops Example

- Lets look at the following simple while loop.

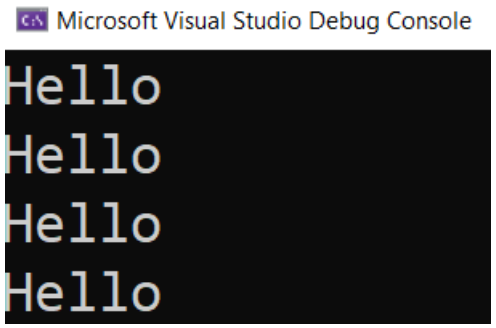
```
int j = 0;
while (j<4) {
    printf("Hello\n");
    j++;
}
```

- This code will print “Hello” to the screen 4 times.

While Loops Example

- See the output of this code:

```
int j = 0;
while (j<4) {
    printf("Hello\n");
    j++;
}
```



- Of course don’t forget # include and void main!

Loops Example

- Remember the problem from earlier:
- How would I write a program that would do the following 3 times **without using loops**.
  - Read in two numbers.
  - Add them together.
  - Print the result.

- How would I now do this **using a while loop**?

While Loops Example

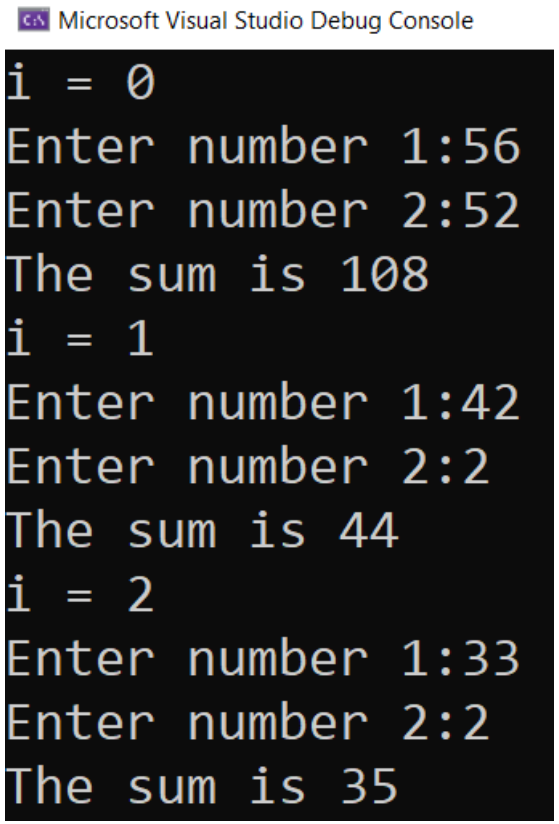
- You would do the following:

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;
    int i = 0;

    while (i<3){
        printf("i = %d\n",i);
        printf("Enter number 1:");
        scanf_s("%d", &num1);
        printf("Enter number 2:");
        scanf_s("%d", &num2);
        total = num1 + num2;
        printf("The sum is %d\n", total);
        i++;
    }
}
```

While Loops Example

- Running the program will give you the following output:

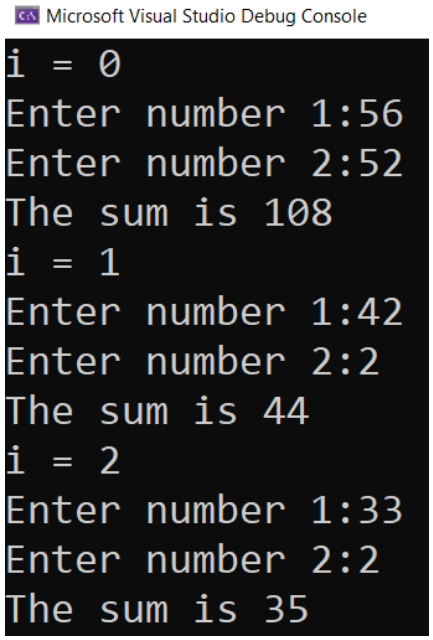
```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;
    int i = 0;

    while (i<3){
        printf("i = %d\n",i);
        printf("Enter number 1:");
        scanf_s("%d", &num1);
        printf("Enter number 2:");
        scanf_s("%d", &num2);
        total = num1 + num2;
        printf("The sum is %d\n", total);
        i++;
    }
}
```



While Loops Example

- This program gives the same output as the previous example (except I am printing an extra line that shows i).

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;
    int i = 0;

    while (i<3){
        printf("i = %d\n",i);
        printf("Enter number 1:");
        scanf_s("%d", &num1);
        printf("Enter number 2:");
        scanf_s("%d", &num2);
        total = num1 + num2;
        printf("The sum is %d\n", total);
        i++;
    }
}
```



Solution Comparison

- The solution on the left uses while loops. The solution on the right does not.

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;
    int i = 0;

    while (i<3){
        printf("i = %d\n",i);
        printf("Enter number 1:");
        scanf_s("%d", &num1);
        printf("Enter number 2:");
        scanf_s("%d", &num2);
        total = num1 + num2;
        printf("The sum is %d\n", total);
        i++;
    }
}
```

```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);

    printf("Enter number 1:");
    scanf_s("%d", &num1);
    printf("Enter number 2:");
    scanf_s("%d", &num2);
    total = num1 + num2;
    printf("The sum is %d\n", total);
}
```

Advantages of While Loops

- Using while loops gives us the following advantages:
  - The code is easier to read.
  - If I want to run the same block of code 10000 times, all I need to do is change the condition to (i<10000).
  - If I want to make a change to multiply numbers together instead of adding them, I only need to do it once!

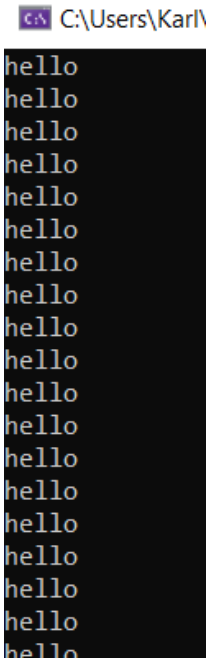
```c
#include <stdio.h>
void main() {
    int num1;
    int num2;
    int total;
    int i = 0;

    while (i<3){
        printf("i = %d\n",i);
        printf("Enter number 1:");
        scanf_s("%d", &num1);
        printf("Enter number 2:");
        scanf_s("%d", &num2);
        total = num1 + num2;
        printf("The sum is %d\n", total);
        i++;
    }
}
```

Avoid infinite loops!

- An infinite loop is a loop that can never end.
- Therefore be careful with your while **condition**.
- You must change a variable inside the while loops body that is used in the condition – otherwise you could end up in an infinite loop.
- Below are both examples of infinite loops:

```
int i = 0;
while (i < 3) {
    printf("hello\n");
}
```

```
while (2 < 3) {
    printf("hello\n");
}
```



DO WHILE LOOPS

Do While Loops

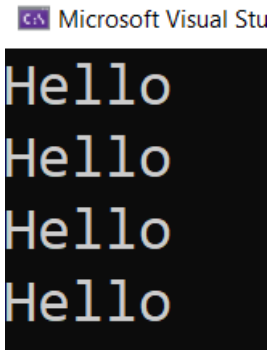
- Do While Loops are a variant of while loops.
- They work in much the same way as while loops.
- Do While Loops have the following structure.

```
do {
    // do something
} while (condition);
```

Do While Loops Example

- Print “Hello” 4 times using a do while loop.

```
int j = 0;
do {
    printf("Hello\n");
    j++;
} while (j<4);
```



Why Use Do While Loops?

- You can use a do while loop if you want to ensure that you execute a block of code **at least once**.

```
int j = 0;
do {
    printf("Hello\n");
    j++;
} while (j<4);
```

"Hello" will be printed at a minimum of once, irrespective of what value j has.

Do While Comparison with While

- Both of the following programs with print “Hello” 4 times.
- The program on the **left** uses a **do while** loop.
- The program on the **right** uses a **while** loop.

```
int j = 0;
do {
    printf("Hello\n");
    j++;
} while (j<4);
```

```
int j = 0;
while (j<4) {
    printf("Hello\n");
    j++;
}
```

EXAMPLE PROBLEM

Example Problem

- Write a program that reads in the users weight in kg and height in meters.
- Calculate their BMI as: BMI = weight / height^2.
- Your program should then give the user an option to go again or to end the program.
- The user should be able to do as many BMI calculations as possible.

Example Problem

- Go to C program solution.

BMI C Program

- The following will code will work:

```c
#include <stdio.h>
void main() {
    float height;
    float weight;
    float bmi;
    int again = 1;
    do
    {
        printf("Enter your weight in kilos: ");
        scanf_s("%f", &weight);

        printf("Enter your height in metres: ");
        scanf_s("%f", &height);

        bmi = (weight) / (height * height);

        printf("Your BMI is: %f.  \n Enter 1 to do again or 0 to exit.\n", bmi);
        scanf_s("%d", &again);
    } while (again == 1);
}
```

BMI C Program Output

- The code will produce the following output: