Chapter 6: Keeping Control
Chapter 6

- The Do-While Statement
- The Switch Statement
- Continue
- Break
- Return and Exit
- Using Recursion
The do-while statement

- The **do-while** statement is another type of loop
- This loop will always be executed at least once
The do-while statement

do
  \textit{statement;} \\
while (expression)

\textbullet\ OR

do
  \{ \\
    \textit{statement block} \\
  \}
while (expression)
The do-while statement

```c
void Example1 (void)
{
    int x=0;
    x=10;

do
{
    x=x-1;
    printf("%d\n",x);
}while(x>5);
}
```
The do-while statement

void Example2 (void) {
    int x=0;
    x=4;

    do {
        x=x-1;
        printf("%d\n",x);
    } while(x>5);
}

3
The Switch Statement

- The `switch` statement provides a way to quickly handle multiple possible values for an expression.
- The case value must be a constant expression.
- Anything unsolved will call the default statement.
- A switch statement begins at its jump point and continues through its cases.
The Switch Statement

switch(expression1)
{
    case c1:
        statement;

    case c2:
        statement;

    default:
        statement;
}

The Switch Statement

```c
void Example3(void) {
    int test_value=0;
    test_value=100;
    switch(test_value) {
        case 1 : printf("Alpha\n");
        case 2 : printf("Beta\n");
        case 3 : printf("Gamma\n");
        default: printf("Omega\n");
    }
}
```
The Continue Statement

- The `continue` statement occurs within a loop.

```
continue;
```

- If `continue` is called, the loop jumps to the beginning of the loop.
- The values of the loop control are not affected.
Continue in a For Loop

```c
printf("For Loop Continue:\n\n");
for(LoopControl=0;
    LoopControl<5;
    LoopControl++)
{
    if (LoopControl%2)
        continue;
    printf("%d\n",LoopControl);
}
printf("\n\n");```

For Loop Continue:
0
2
4
Continue in a While Loop

```c
printf("While Loop Continue:\n");
LoopControl=0;
while(LoopControl<5)
{
    LoopControl++;
    if (!LoopControl%2)
        continue;

    printf("%d\n",LoopControl);
}
printf("\n")
```

While Loop Continue:
2
4
printf("DoWhile Loop Continue:\n");
LoopControl=0;
do
{
    LoopControl++;
    if (LoopControl%2)
        continue;
    printf("%d\n ",LoopControl);
}
while(LoopControl<5);
printf("\n");
More Continue

- What will this code do?

```c
void Example5(void)
{
    int LoopControl=0;
    printf("While Loop Continue:\n");
    while(LoopControl<5)
    {
        if (LoopControl%2)  
            continue;
        printf("%d ",LoopControl);
        LoopControl++;
    }
    printf("\n");
}
```

While Loop Continue: 0

*It will print out the first value, and then loop forever because the continue blocks the increment once it reaches 1.*
The Break Statement

- The **break** statement alters the flow of control for a loop.

```c
break;
```

- The break statement stops the current loop and jumps to the end of the loop.
- The values of the loop control are not affected.
int LoopControl=0;
printf("For Loop Break:\n");
for (LoopControl=0;
    LoopControl<10;
    LoopControl++)
{
    if (LoopControl/4)
        break;
    printf("%d\n",LoopControl);
}
printf("\n");

For Loop Break:
0
1
2
3
printf("While Loop Break:\n");
LoopControl=56;
while (LoopControl<70) {
    LoopControl++;
    if (LoopControl/60)
        break;
    printf("\%d\n", LoopControl)
} printf("\n");

While Loop Break:
57
58
59
printf("DoWhile Loop Break:\n");

int LoopControl=75;
do
{
    LoopControl++;
    if (LoopControl/80)
        break;
    printf("%d\n",LoopControl-80);
}
while(LoopControl<100);
printf("\n");
Break and Switch

• The `break` statement can also alter the behavior of a switch statement.
Break and Switch Statement

test_value=3;
switch(test_value)
{
    case 1 :
        printf("Alpha\n");
        break;
    case 2 :
        printf("Beta\n");
        break;
    case 3 :
        printf("Gamma\n");
    default:
        printf("Omega\n");
        break;
}
Ending a program: Return and Exit

• As before, return ends a function.
• return; can send ‘void’ or nothing for a function with a void return type back to the calling function.
• return expression; can send a value for a function with a return type back to the calling function.
• Return can only end a program if it is called in the main.
Ending a program: Return and Exit

- An **exit** statement ends the current program, and can take a parameter of type int.
- exit can terminate a program from anywhere in its function lists.
- This value is returned to the calling environment.
- Comes from the standard library, use `#include <stdlib.h>` at the beginning of a program.
int LoopControl=0;
printf("Begin Routine\n");
for(LoopControl=0;
    LoopControl<5;
    LoopControl++)
{
    printf("%d\n",LoopControl);
    if(LoopControl==3)
        exit(0);
}
printf("End of Routine\n");
Using Recursion

• **Recursion** occurs when a subprogram calls itself.

```c
int MyRecursion(int MyData)
{
    int result=0;
    result=MyRecursion(MyData);
    return (result);
}
```
Using Recursion

- In mathematics, recursion occurs when the next calculated value directly depends on the previous value.
- \( x_n = \sum x_{(i-1)} + K \)
- \( x_0 = K \)
- \( x_1 = x_0 + K \)
- \( x_2 = x_1 + x_0 + K \)
- \( x_3 = x_2 + x_1 + K \)
Using Recursion

- A base to a power:
  \[ x^n = x \times x^{(n-1)}, \quad x^0 = 1 \]

- A factorial
  \[ x! = x \times (x-1)!, \quad 0! = 1 \]
Using Recursion

```c
int Factorial(int n)
{
    if(n==0)
        return 1;
    else
        return n*Factorial(n-1);
}
```
Using Recursion

int Factorial(int n) {
    if(n==0)
        return 1;
    else
        return n*Factorial(n-1);
}

printf("0! = %d\n",Factorial(0));
printf("1! = %d\n",Factorial(1));
printf("2! = %d\n",Factorial(2));
printf("3! = %d\n",Factorial(3));
printf("4! = %d\n",Factorial(4));

0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
Using Recursion

int FindPower(int base, int power)
{
    if (power==0)
        return 1;
    else
        return base*FindPower(base, power-1);
}
Using Recursion

```c
int FindPower(int base, int power)
{
    if (power==0)
        return 1;
    else
        return base * FindPower(base, power-1);
}
```

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$