**Lab Sections**

1. Objectives
2. Introduction
3. Definitions
4. Declaration Syntax
5. Experiments

|  |
| --- |
|  |

Recursion

**Recursion**

1. **Objectives**

**After this experiment you will be able to implement a simple recursive function.**

1. **Introduction**

There are two types of recursion, direct and indirect. Direct recursion occurs when a function calls itself. Indirect recursion occurs when a function calls another function, which calls another function, which calls another function, until eventually the original function is called. Many solutions to problems are easily implemented recursively.

All recursive algorithms divide a large problem into smaller less complex sub-problems. This process is known as “Divide-and-conquer”. Each sub-problem is also divided into smaller sub-problems until eventually one of those sub-problems can be solved. This solution is then returned to its predecessor (parent problem), which uses this solution to solve its problem. This procedure is continued until all sub-problems have been solved. Once this occurs, the solution to the original problem can be determined. Each recursive call represents a new totally independent call from all the previous function calls. Memory must be allocated for its formal parameters, return address and so forth.

All recursive algorithms can be implemented iteratively. Iteratively means they can be implemented using loops (while, for, etc.).

1. **Definitions/Important Terms**

We will define several terms you need to know to understand recursion. They are as follows:

1. A **base, sometimes referred to as the anchor or trivial case,** is a very easy solution that does not require too much thought to solve.
2. **Recursive case** is the step in the algorithm where the problem is divided into smaller less complex problems.
3. **Syntax for a Recursive Function**

Following is a simple “multiple-alternative if” statement that can be used to implement a recursive function. However, please keep in mind that recursive functions may have different formats. To write a recursive function usually requires two steps. First, identify the base case. Secondly, identify any recursive cases.

if (base case)

return trivial solution

else //(recursive case 1)

return (solution from recursive call)

.....

else //(recursive call)

return (solution from recursive call)

More information on recursion can be found in your course textbook and on the web.

1. **Experiments**

**Step 1: In this experiment you will learn how to declare functions.**

**Enter, save, compile and execute the following program in MSVS. Call the new directory “recursionExp1” and the program “recursion1.cpp”. Answer the questions below:**

#include <iostream>

using namespace std;

void recursive\_countdown(int count)

{

if (count == 0)

cout<<"count="<<count<<endl;

else

{

cout<<"count="<<count<<endl;

recursive\_countdown(--count);

}

}

int main(void)

{

int count = 10;

recursive\_countdown(count);

return 0;

}

1. What does the program do?
2. Please write the base case?
3. How many times is “recursive\_countdown” called recursively?
4. Please rewrite the program in Step 1 using an iterative function to countdown? Call the function “iterative\_countdown”.
5. Which type of function (recursive or iterative) executes the fastest? Please explain?

**Step 2: In this experiment you will learn how to declare functions.**

**Enter, save, compile and execute the following program in MSVS. Call the new directory**

**“recursionExp2” and the program “recursion2.cpp”. Answer the question below:**

#include <iostream>

using namespace std;

int R\_power(int count, const int & base)

{

if (count == 0)

return 1;

else

return base \* R\_power(count-1,base);

}

int I\_power(int count, const int & base)

{

int multiend = 1;

while (count > 0)

{

multiend \*= base;

count--;

}

return multiend;

}

int main()

{

int count = 10;

int base = 2;

cout<<R\_power(count,base)<<endl;

cout<<I\_power(count,base)<<endl;

return 0;

}

1. Please explain the operation of the functions “R\_power” and “I\_Power”?

**Step 3: In this experiment you will learn how to declare functions.**

**Enter, save, compile and execute the following program in MSVS. Call the new directory**

**“recursionExp3” and the program “recursion3.cpp”. Answer the questions below:**

1. Please write an iterative function to reverse a string? For example, if a string is “hello” then the function will return “olleh”.
2. Please write a recursive function to reverse a string? For example, if a string is “hello” then the function will return “olleh”.