Function Calls

Lab Sections

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

Function Calls

1. Objectives

After you complete this experiment you will be able to:

a. **Understand input and output parameters**
b. **compare and contrast call-by-value and call-by-reference**

2. Introduction

During a function call memory is allocated for the formal parameters and local variables. Memory is also allocated for the address of the instruction to return to after the function has executed. There are two ways arguments are passed to a function during a function call, call-by-value and call-by-reference. Being able to call functions has many advantages which include: 1) allows us to write complex programs; 2) allows the user to decompose a program into smaller manageable components; 3) makes it easier to find errors; 4) makes it easier to maintain the program.

3. Definitions & Important Terms

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

1. **Actual arguments** are the arguments used in the function call statement.
2. **Formal parameters** are the parameters used in the function header.
3. During a **call-by-value** a copy of the actual argument is made and placed into its corresponding formal parameter.
4. During a **call-by-reference** the address of the actual argument is copied into the formal parameter, when pointers are used.
5. During a **call-by-reference** the formal parameter is an alias for the actual argument when a reference is used in C++.
6. A formal parameter is called an **output parameter** when it is passed-by-reference. A formal parameter is called an **input parameter** if it is passed-by-value
7. If the value of a formal parameter is changed inside a function, and if this changed value is needed outside the function, it should be labeled as an output parameter or as a parameter that is passed by reference.
8. If the value of a formal parameter is only used inside a function, it should be called an input parameter or a parameter that is passed by value.
9. A **prototype** is a function declaration.
10. **References** are implemented internally as pointers

4. Declaration Syntax

```
return_type  function_name(formal_parameter_list);
```

Example:

```
void input(int &a, double c, char * m);
```

Explanation: a and m are passed-by-reference, and c is passed-by-value.

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

5. Experiments

Step 1: In this experiment you will investigate the call-by-value mechanism.
Enter, save, compile and execute the following program in MSVS. Call the new project "FunctionCallsExp1" and the program "FunctionCalls1.cpp". Answer the questions below:

```
#include <iostream>
#include <string>

using namespace std;

void swap(int a, int b)
{
	int temp = a;
	cout<<"a and b have the following values at the start of swap"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	a=b;
	b=temp;
	cout<<"a and b have the following values at the end of swap"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;

}
int main()
{
	int a=111, b=222;

	cout<<"a and b have the following values before swap is called"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	swap(a,b);
	cout<<"a and b have the following values after swap is called"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	return 0;
}
```

Question 1: Are the variables a and b used in the function main the same a and b variables that are used in the function swap in the program in Step 1? Explain your answer.

Question 2: Were the values stored in the variables a and b in the function main swapped after the execution of the function swap in the program in Step 1? Explain your observations.

Step 2: In this experiment you will investigate the call-by-reference mechanism.

Execute the following program and answer the questions below:

```
#include <iostream>
#include <string>

using namespace std;

void swap(int & a, int & b)
{
	int temp = a;
	cout<<"a and b have the following values at the start of swap"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	a=b;
	b=temp;
	cout<<"a and b have the following values at the end of swap"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;

}
int main()
{
	int a=111, b=222;

	cout<<"a and b have the following values before swap is called"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	swap(a,b);
	cout<<"a and b have the following values after swap is called"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	return 0;
}
```

Question 1: Are the variables a and b used in the function main the same a and b variables that are used in the function swap in the program in Step 2? Please explain your answer.

Question 2: Were the values stored in the variables a and b in the function main swapped after the execution of the function swap in the program in Step 2? Explain your observations.

Step 3: In this experiment you investigate the call-by-reference mechanism that uses pointers.

Execute the following program and answer the questions below:

```
#include <iostream>
#include <string>

using namespace std;

void swap(int * a, int * b)
{
	int temp = *a;
	cout<<"the contents of the memories pointed to by a and b have the following
values at the start of swap"<<endl;
	cout<<"*a = "<<*a<<" and *b = "<<*b<<endl;
	*a=*b;
	*b=temp;
	cout<<"the contents of the memories pointed to by a and b have the following
values at the end of swap"<<endl;
	cout<<"*a = "<<*a<<" and *b = "<<*b<<endl;

}
int main()
{
	int a=111, b=222;

	cout<<"a and b have the following values before swap is called"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	swap(&a,&b);
	cout<<"a and b have the following values after swap is called"<<endl;
	cout<<"a = "<<a<<" and b = "<<b<<endl;
	return 0;
}
```

Question 1: What is the purpose of the '&' and '*' symbols in the main and swap functions in the program in Step 3? Explain your answer.

Question 2: Are the variables a and b used in the function main the same a and b variables that are used in the function swap in the program in Step 3? Explain your answer.

Question 3: Were the values stored in the variables a and b in the function main swapped after the execution of the function swap in the program in Step 3? Explain your observations.

Question 4: What types of function invocations were used in the programs in Steps 1, 2 and 3?