Bangladesh University of Engineering & Technology(BUET)

Assignment On:

A Comparative Analysis Between Function Overloading and Default Arguments

Submitted To:

Dr. Tanzima Hashem

Professor

Department of Computer Science & Engineering, BUET

Submitted By:

Syed Jarullah Hisham

Roll – 1805004

CSE ’18 Section A

Date: 09-09-2020

A Comparative Analysis Between Function Overloading and Default Arguments

Function Overloading:

Function overloading denotes the process of having two or more functions with the same name but difference in parameters which allows the programmer to restate the same function either by using different types of arguments or different number of arguments or both; according to the situation. C++ compiler distinguishes the functions at the compilation time.

Default Arguments:

Default argument is an argument to the function which the programmer may not specify at the time of calling the function. The compiler automatically adds those default values. However, there are some rules for specifying default arguments.

- Firstly, default arguments must be specified from the right of the parameter list.
- Default arguments should not be local variables
- Default arguments must be either constants or global variables

Comparative Analysis:

Though function overloading and default arguments are closely related, there are some differences in usages, advantages, disadvantages. So, here is a brief description about the comparison of these two features in various context.

- **Simple function:**

  First, we start looking at the very basic usages of these two features.

Overloading By differing the types of arguments & number of arguments

```
int area (int a,int b)
{   return a*b;   }
float area (float a)
{    return 3.14*a*a;   }
float area (float a,float b)
{   return a*b;   }
double area (double a,double b)
{    return a*b;  }
```

area(3,4)	**[area(int,int)]**
area(4)	**[area(float)]**
area(2.3f,4.5f)	**[area(float,float)]**
area(2.9,3.4)	**[area(double,double)]**

the usage of default arguments

```
double areaTri (double a,double b = 1,
double c = 1)
{
   double p=(a+b+c)/2.0;
   double  s=sqrt (p*(p-a)*(p-b)*(p-c));
   return s;
}
```

areaTri(1.5)	**[a=1.5,b=1(default),c=1(default)]**
areaTri(1.5,1.5)	**[a=1.5,b=1.5,c=1(default)]**
areaTri(2,2,2)	**[a=2,b=2,c=2 where no default]**

- **Invalid Cases:**

  There are some invalid usages of both function overloading and default arguments. For example,

/* Function overloading */	/* Default argument */

`int sum(int a,int b);`
`float sum(int a,int b);`
	`int sum(int a=10,int b,int c=20);`
	`int sum(int a,int b=20,int c);`
	`int sum(int a=10,int b=20,int c);`

these two functions are not overloaded. Difference in return type only can't make functions overloaded.	All these three functions declaration invalid because default arguments must be declared from right to left.

- **Class member function:**

  Function overloading and default arguments are also used in class member functions having almost the same characteristics as that of the simple function. For example,

/* Function overloading*/	/* Default argument*/

`class Profile`
`{`
`public:`
`string name;`
`int id;`
`void setProfile (string name) {`
`this->name=name;`
`this->id=-1;`
`}`
`void setProfile (string name,int id) {`
`this->name=name;`
`this->id=id;`
`}`
`};`
`Profile p;`
`p.setProfile("Hisham");` **[Hisham -1]**
`p.setProfile("Hisham",12);` **[Hisham 12]**
	`/* ----------*/`
	`void setProfile (string name,int id=-1) {`
	`this->name=name;`
	`this->id=id;`
	`}`
	`/*----------*/`

- **Constructor:**

  We can overload constructor function in a class and even we can use default argument there. The rule is same as class member function. For example,

/* Function overloading*/	/* Default argument */

`/* In previous example/`
`Profile(string name){`
`----------`
`}`
`Profile(string name,int id){`
`----------`
`}`
	`/ *----------* /`
	`Profile(string name,int id=-1){`
	`----------`
	`}`

- **Copy Constructor:**

  Function overloading and Default argument play a crucial role at the time of copy constructor too. Frequently, we have to initialize an object from another object of the same type where we need copy constructor. For example,

/* Function overloading*/	/* Default argument */

/*class Profile*/
 	
Profile(const Profile &p){
this->name=p.name;
this->id=p.id;
}
Profile q("Hisham",12);
Profile p=q;
 	
**[p.name=Hisham
p.id=12]**
	/* ---------- */
	
	Profile(const Profile &p,int id=-1){
	this->name=p.name;
	this->id=id;
	}
	Profile q("Hisham");
	
	/* same as function overloading/

- **Recursive Function:**

  In the case of recursive function also we can use these two features to meet our special purpose. Again, more than one function at function overloading but only one function at default argument will accomplish our purpose. For example;

/ * Function Overloading*/	/ * default argument */

long fact(int n)
{
if(n==1) return 1;
return n*fact(n-1);
}
long fact(int n,int c)
{
if(c==0) return 1;
return n*fact(n-1,c-1);
}
	long fact(int n,int c=10)
	{
	if(c==0 || n==0) return 1;
	return n*fact(n-1,c-1);
	}

Now, if we call fact(10) and fact(10,2) ; both code will show the output 362800 and 90 ; which is a great example to show the usage of function overloading and default argument at recursive function.

- **Ambiguity:**

  Sometimes, function overloading and default arguments can cause ambiguity in case of not properly being used.

/ * Function Overloading*/	/ * default argument*/

int area (int a,int b) {.....}
float area (float a,float b) {.....}
 	
area (2,3); // unambiguous
area (2.3,3.4); // ambiguous
	float area (int r) {...}
	float area (int a,int b=1) {...}
	
	area (12,23); // unambiguous
	area (10); // ambiguous

In the first example, area(2,3) will call area(int,int) function but when area(2.3,3.4) is called ;It will show-“call for area(double,double) is ambiguous”. In the next example,area(12,23) will call the overloaded function which has a default argument. But in the second function, compiler will not be able to figure out which function to call.

Summary:

However, there are various other topics like-template function, function overriding etc. where these two features play crucial role. In conclusion, function overloading and default argument being closely related to each other, in case of their usages, both similarity & difference exists having different advantages and flexibility in different situations which creates a large area of significance and impact in computer science.