Overloading functions

Functions with same name but with different parameters, different number of parameters and parameters in different order are said to be overloaded.

```
1   #include <iostream>
2   using namespace std;
3
4   float divide (float p){ return (p/2); }
5   int divide (float p){ return (p/2); }
6
7   int main(){
8       float a = 5.0, b = 3.0;
9       cout << divide(a) << endl;
10  }
```

Compiler does not distinguish functions by return data type

```
ogs & others
Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck/Vera++

File              Line   Message
                         === Build file: "no target" in "no project" (compiler: unknown) ===
F:\CHN-103\L15...        In function 'int divide(float)':
F:\CHN-103\L15...  5     error: ambiguating new declaration of 'int divide(float)'
F:\CHN-103\L15...  4     note: old declaration 'float divide(float)'
                         === Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Overloading functions

```
1   #include <iostream>
2   using namespace std;
3
4   float divide (float p){
5       cout << "Single parameter function called\n";
6       return (p/2);
7   }
8   float divide (float p, float q){
9       cout << "Two parameter function called\n";
10      return (p/q);
11  }
12
13  int main(){
14      float a = 5.0, b = 3.0;
15      cout << divide(a) << endl;
16      cout << divide(a,b) << endl;
17  }
```

F:\CHN-103\L15_Functions\funcDefault.exe

```
Single parameter function called
2.5
Two parameter function called
1.66667

Process returned 0 (0x0)   execution time : 0.219 s
Press any key to continue.
```

Overloading functions

```
1   #include <iostream>
2   using namespace std;
3
4   float divide (float p, int q){
5       cout << "float first and int next\n";
6       return (p/q);
7   }
8   float divide (int q, float p){
9       cout << "Int first and float next\n";
10      return (p/q);
11  }
12
13  int main(){
14      float a = 5.0, b = 3.0;
15      int c = 2;
16      cout << divide(a,c) << endl;
17      cout << divide(c,a) << endl;
18  }
19
```

F:\CHN-103\L15_Functions\funcOverloadingNumParameters.exe

```
float first and int next
2.5
Int first and float next
2.5

Process returned 0 (0x0)   execution time : 0.156 s
Press any key to continue.
```

Overloading functions

```
1   #include <iostream>
2   using namespace std;
3
4   float divide (float p, int q){
5       cout << "float first and int next\n";
6       return (p/q);
7   }
8   /*float divide (int q, float p){
9       cout << "Int first and float next\n";
10      return (p/q);
11  }*/
12  float divide (float p, float q){
13      cout << "Both parameters are float\n";
14      return (p/q);
15  }
16
17  int main(){
18      float a = 5.0, b = 3.0;
19      int c = 2;
20      cout << divide(a,c) << endl;
21      cout << divide(c,a) << endl;
22  }
23
```

Implicit promotion is applied by compiler for second call.

F:\CHN-103\L15_Functions\funcOverloadingNumParameters.exe

```
float first and int next
2.5
Both parameters are float
0.4

Process returned 0 (0x0)   execution time : 0.156 s
Press any key to continue.
```

Overloading functions

```
1   #include <iostream>
2   using namespace std;
3
4   float divide (float p, int q){
5       cout << "float first and int next\n";
6       return (p/q);
7   }
8   /*float divide (int q, float p){
9       cout << "Int first and float next\n";
10      return (p/q);
11  }*/
12  /*float divide (float p, float q){
13      cout << "Both parameters are float\n";
14      return (p/q);
15  }*/
16  float divide (int p, int q){
17      cout << "Both parameters are int\n";
18      return ((float)p/q);
19  }
20
21  int main(){
22      float a = 5.5, b = 3.0;
23      int c = 2;
24      cout << divide(a,c) << endl;
25      cout << divide(c,a) << endl;
26  }
```

The compiler tries to match datatypes left to right. Caution!! here there is loss of accuracy because 'a' is demoted to ***int*** datatype before being promoted to ***float*** for calculation.

```
F:\CHN-103\L15_Functions\funcOverloadingNumParameters.exe
float first and int next
2.75
Both parameters are int
0.4

Process returned 0 (0x0)   execution time : 0.122 s
Press any key to continue.
```

Overloading functions

```
1   #include <iostream>
2   using namespace std;
3
4   float divide (float p, int q = 2){
5       return (p/q);
6   }
7
8   int main(){
9       float a = 5.5;
10      int c = 3;
11      cout << divide(a) << endl;
12      cout << divide(a,c) << endl;
13  }
```

Parameters can be given default values such as q = 2, starting from right to left.

F:\CHN-103\L15_Functions\funcDefault.exe

```
2.75
1.83333

Process returned 0 (0x0)   execution time : 0.141 s
Press any key to continue.
```

Overloading functions

```
1   #include <iostream>
2   using namespace std;
3   float divide (float p, int q = 2){
4       return (p/q);
5   }
6   float divide (float p){
7       return (p/2);
8   }
9   int main(){
10      float a = 5.5;
11      int c = 3;
12      cout << divide(a) << endl;
13      cout << divide(a,c) << endl;
14  }
```

Caution!! Default arguments may cause ambiguity

ogs & others

Code::Blocks | Search results | Cccc | Build log | **Build messages** | CppCheck/Vera++

File	Line	Message
		=== Build file: "no target" in "no project" (compiler: unknown) ===
F:\CHN-103\L15...		In function 'int main()':
F:\CHN-103\L15...	15	error: call of overloaded 'divide(float&)' is ambiguous
F:\CHN-103\L15...	4	note: candidate: float divide(float, int)
F:\CHN-103\L15...	8	note: candidate: float divide(float)
		=== Build failed: 1 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===

Functions and Arrays

```
1   #include <iostream>
2   #include <iomanip>
3   #include <cmath>
4
5   using namespace std;
6
7   int main(){
8
9       int m,n;
10      cout << "Enter value of m: "; cin >> m;
11      cout << "Enter value of n: "; cin >> n;
12
13      // Allocating memory for matrix A of size m x n
14      double **A;
15      A = new double*[m];
16      for (int i = 0; i < m; i++){
17          A[i] = new double[n];
18      }
19
20      // Allocating memory for matrix B of size m x n
21      double **B;
22      B = new double*[m];
23      for (int i = 0; i < m; i++){
24          B[i] = new double[n];
25      }
```

Allocating block

Allocating block

Functions and Arrays

```
26
27  // Get values of elements
28  for (int i = 0; i < m; i++)
29      for (int j = 0; j < m; j++){
30          cout << "A[" << i << "][" << j << "]: ";
31          cin >> A[i][j];
32      }
33
34  // Get values of elements
35  for (int i = 0; i < m; i++)
36      for (int j = 0; j < m; j++){
37          cout << "B[" << i << "][" << j << "]: ";
38          cin >> B[i][j];
39      }
40
41  // Allocating memory for matrix B of size m x n
42  double **C;
43  C = new double*[m];
44  for (int i = 0; i < m; i++){
45      C[i] = new double[n];
46  }
```

Data Input block

Data Input block

Allocating block

Functions and Arrays

```
47
48      // Get values of elements
49      for (int i = 0; i < m; i++)
50          for (int j = 0; j < m; j++){
51              C[i][j] = A[i][j] + B[i][j];
52          }
53
54      // Display values of elements
55      for (int i = 0; i < m; i++){
56          for (int j = 0; j < m; j++){
57              cout << setw(3) << C[i][j];
58          }
59          cout << endl;
60      }
61  }
```

Computing block

Display block

```
F:\CHN-103\L15_Functions\funcAr
Enter value of m: 3
Enter value of n: 3
A[0][0]: 1
A[0][1]: 2
A[0][2]: 3
A[1][0]: 4
A[1][1]: 5
A[1][2]: 6
A[2][0]: 7
A[2][1]: 8
A[2][2]: 9
B[0][0]: 1
B[0][1]: 0
B[0][2]: 0
B[1][0]: 0
B[1][1]: 1
B[1][2]: 0
B[2][0]: 0
B[2][1]: 0
B[2][2]: 1
  2  2  3
  4  6  6
  7  8 10

Process returned 0 (0x0)
Press any key to continue.
```

We can put all the distinct blocks into modules of functions.

Functions and Arrays

Function declarations for implementing each block

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

double ** allocateMatrix(int m, int n);
void getMatrix(double **A, int m, int n);
void addMatrix(double **A, double **B, double **C, int m, int n);
void displayMatrix(double **A, int m, int n);
```

Allocating block

Get input block

Compute block

Display block

Functions and Arrays

```
12  int main(){
13      int m,n;
14      cout << "Enter value of m: "; cin >> m;
15      cout << "Enter value of n: "; cin >> n;
16      // Allocating memory for matrix A of size m x n
17      double **A; A = allocateMatrix(m,n);
18      // Allocating memory for matrix B of size m x n
19      double **B; B = allocateMatrix(m,n);
20      // Get values of elements
21      getMatrix(A,m,n);
22      getMatrix(B,m,n);
23      // Allocating memory for matrix B of size m x n
24      double **C; C = allocateMatrix(m,n);
25      // Get values of elements
26      addMatrix(A,B,C,m,n);
27      // Display values of elements
28      displayMatrix(C,m,n);
29  }
```

Functions and Arrays

```
double** allocateMatrix(int m, int n){
    double **A;
    A = new double*[m];
    for (int i = 0; i < m; i++){
        A[i] = new double[n];
    }
    return (A);
}
```

```
void getMatrix(double **A, int m, int n){
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++){
            cout << "[" << i << "][" << j << "]: ";
            cin >> A[i][j];
        }
}
```

Functions and Arrays

```
48  void addMatrix(double **A, double **B, double **C, int m, int n){
49      for (int i = 0; i < m; i++)
50          for (int j = 0; j < n; j++){
51              C[i][j] = A[i][j] + B[i][j];
52          }
53  }
```

```
55  void displayMatrix(double **A, int m, int n){
56      for (int i = 0; i < m; i++){
57          for (int j = 0; j < n; j++){
58              cout << setw(3) << A[i][j];
59          }
60          cout << endl;
61      }
62  }
```

Function pointers

```
1   #include <iostream>
2   using namespace std;
3   double addNumbers(double, double);
4   double multiplyNumbers(double, double);
5
6   int main(){
7       double (*fptr) (double, double);
8       short choice;
9       cout << "Do you want to 1. add or 2. multiply? Enter 1 or 2: ";
10      cin >> choice;
11
12      switch (choice){
13          case 1:
14              fptr = &addNumbers;
15              break;
16          case 2:
17              fptr = &multiplyNumbers;
18              break;
19          default:
20              cout << "Choice not available!!";
21      }
22      double a, b;
23      cout << "Enter a: "; cin >> a;
24      cout << "Enter b: "; cin >> b;
25      cout << "Result is " << (*fptr)(a,b) << endl;
26  }
```

Function pointer declaration

Function pointer Assignment

Function pointer Dereferencing

Function pointers

```
28  double addNumbers(double a, double b){
29      return (a + b);
30  }
31
32  double multiplyNumbers(double a, double b){
33      return ( a * b );
34  }
35
```

F:\CHN-103\L15_Functions\funcPointer.exe

```
Do you want to 1. add or 2. multiply? Enter 1 or 2: 1
Enter a: 2
Enter b: 3
Result is 5

Process returned 0 (0x0)   execution time : 4.853 s
Press any key to continue.
```

F:\CHN-103\L15_Functions\funcPointer.exe

```
Do you want to 1. add or 2. multiply? Enter 1 or 2: 2
Enter a: 2
Enter b: 3
Result is 6

Process returned 0 (0x0)   execution time : 4.906 s
Press any key to continue.
```

Inline function

```
G:\CHN-103\L15_Functions\funcInline.exe
######
@@@@@@

Process returned 0 (0x0)   execution time : 0.109 s
Press any key to continue.
```

```cpp
#include <iostream>
#define MAX(X,Y) (X) > (Y)? (X) : (Y)

const int N = 10;
using namespace std;

inline void printline(char ch, int n){
    for (int i = 0; i < n; i++)
        cout << ch;
}

inline double max(double a, double b){
    return (a > b ? a : b);
}

int main(){

    double a = 3.4, b = 6.7;
    printline('#',int (MAX(a,b)));
    cout << endl;
    printline('@',int (max(a,b)));
    cout << endl;

}
```

Inline keyword is used to indicate that a function can be inserted inline, however it is up to the compiler to insert it or not.

The macro MAX is preprocessed by preprocessor and will definitely be replaced wherever it appears in the code.

Library functions - <cctype>

Prototype	Description
int *isdigit* (int c)	Returns 1 if c is a digit and 0 otherwise
int *isalpha* (int c)	Returns 1 if c is a letter and 0 otherwise
int *isalnum* (int c)	Returns 1 if c is a letter or a digit and 0 otherwise
int *isxdigit* (int c)	Returns 1 if c is a hexadecimal digit character (a-z, A-Z or 0-9) and 0 otherwise
int *islower* (int c)	Returns 1 if c is a lowercase letter (a-z) and 0 otherwise
int *isupper* (int c)	Returns 1 if c is a uppercase letter (A-Z) and 0 otherwise
int *tolower* (int c)	Changes uppercase letter to lowercase, remaining characters do not change.
int *toupper* (int c)	Changes lowercase letter to uppercase, remaining characters do not change.

Library functions - <cctype>

Prototype	Description
int *isspace* (int c)	Returns 1 if c is a whitespace (‘\n’,’ ’,’\t’,’\f’,’\r’,’\v’) and 0 otherwise
int *iscntrl* (int c)	Returns 1 if c is a control character (‘\n’,’\f’,’\r’,’\t’,’\v’,’\a’,’\b’) and 0 otherwise
int *ispunct* (int c)	Returns 1 if c is a printing character other than space, digit, or letter and 0 otherwise
int *isprint* (int c)	Returns 1 if c is a printing character including space (‘ ‘) and 0 otherwise
int *isgraph* (int c)	Returns 1 if c is a printing character other than space and 0 otherwise

Library functions - <cctype>

```
1   #include <iostream>
2   #include <iomanip>
3   #include <fstream>
4   #include <cctype>
5   using namespace std;
6
7   int main(){
8
9       ifstream inFile("Sample.txt");
10      if (inFile.is_open()){
11          cout << "File opened successfully.\n";
12          char ch;
13          short nLines = 0;
14          short nDigit = 0;
15          short nAlpha = 0;
16          short nAlnum = 0;
17          short nLower = 0;
18          short nUpper = 0;
19          short nSpace = 0;
20          short nCntrl = 0;
21          short nPunct = 0;
22          short nPrint = 0;
23          short nGraph = 0;
```

Library functions - <cctype>

```
24          while (!inFile.eof()){
25              short nStr = 1;
26              while ((ch = inFile.get()) != '\n' && ch != EOF){
27                  if (ch == ' ' || ch == '\t') nStr++;
28                  if (isdigit(ch)) nDigit++;
29                  if (isalpha(ch)) nAlpha++;
30                  if (isalnum(ch)) nAlnum++;
31                  if (islower(ch)) nLower++;
32                  if (isupper(ch)) nUpper++;
33                  if (isspace(ch)) nSpace++;
34                  if (iscntrl(ch)) nCntrl++;
35                  if (ispunct(ch)) nPunct++;
36                  if (isprint(ch)) nPrint++;
37                  if (isgraph(ch)) nGraph++;
38              }
39              nLines++;
40              cout << "Number of strings in line["
41                   << setw(3) << nLines << "] = " << nStr << endl;
42          }
43          inFile.close();
```

Library functions - <cctype>

```
44
45          cout << "File stats:\n";
46          cout << "Number of digits = " << nDigit << endl;
47          cout << "Number of alphabets = " << nAlpha << endl;
48          cout << "Number of digits + alphabets = " << nAlnum << endl;
49          cout << "Number of lowercase letters = " << nLower << endl;
50          cout << "Number of uppercase letters = " << nUpper << endl;
51          cout << "Number of spaces = " << nSpace << endl;
52          cout << "Number of control characters = " << nCntrl << endl;
53          cout << "Number of punctuations = " << nPunct << endl;
54          cout << "Number of printing characters (including spaces) = "
55               << nPrint << endl;
56          cout << "Number of printing characters (excluding spaces) = "
57               << nGraph << endl;
58      }
59      else{
60          cout << "The file could not be opened.\n";
61      }
62  }
```