Code 1.0

Code1.0

Main.cpp

```cpp
#include "generic.h"
void increment();
void display();
int main() {
    increment();
    display();
    counter++;
    std::cout << " [main]   counter = " << counter << std::endl;
    std::cout << " [main]   Address of counter = " << &counter << std::endl;
}
```

Generic.h

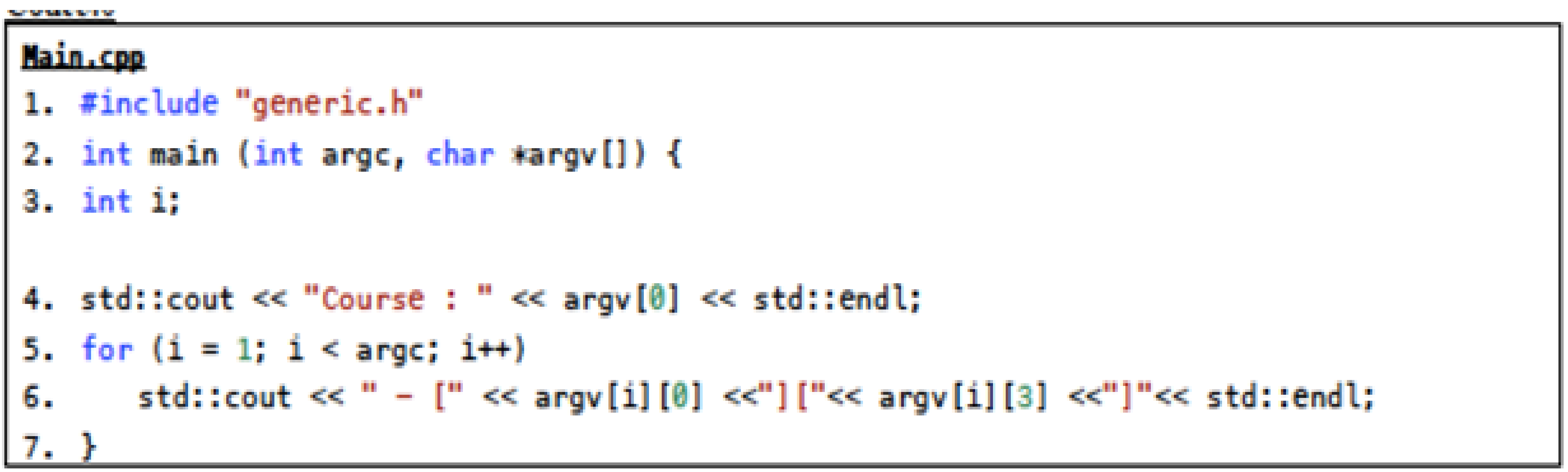
```cpp
#include <iostream>
static int counter = 0;
```

Unit_a.cpp

```cpp
#include "generic.h"
void increment()
{
   int counter = 2;
   counter = counter + 2;
   std::cout << " [increment]   Address of counter = " << &counter << std::endl;
}
void display( )
{
    std::cout << "[display]   counter = " << counter << std::endl;
    std::cout << "[display]   Address of counter = " << &counter << std::endl;
}
```

Code 3.0

Code3.0



Main.cpp

```cpp
#include "generic.h"
int main (int argc, char *argv[]) {
int i;

std::cout << "Course : " << argv[0] << std::endl;
for (i = 1; i < argc; i++)
    std::cout << " - [" << argv[i][0] <<"]["<< argv[i][3] <<"]"<< std::endl;
}
```

Assume the following command line arguments are passed to the program in Code 3.0
Assignments Workshops Tests Exam

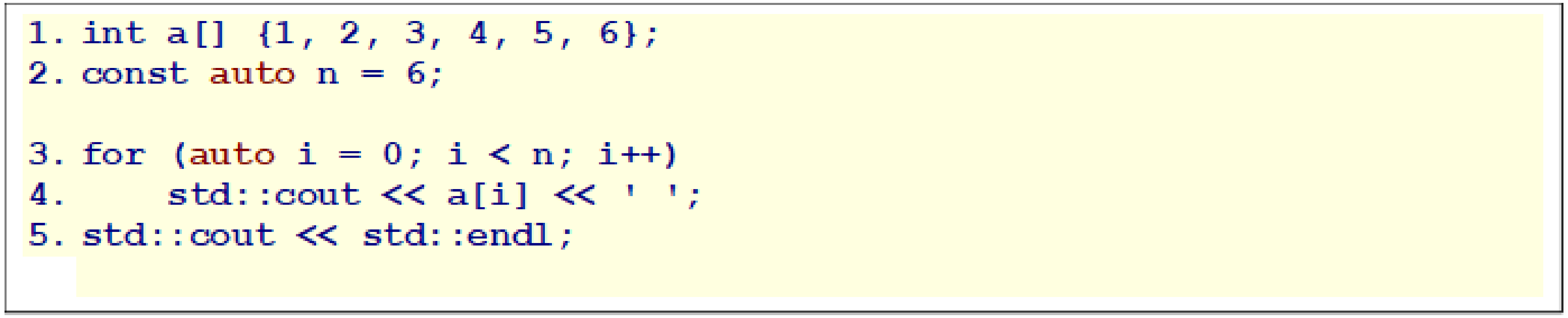
Code 4.0

```cpp
unsigned char x = 0;
unsigned char y = 150;
std::cout << " Entering the loop " << std::endl;
for ( ; x < 2*y; x++ )
{
    std::cout << " x = " << (int) x <<  std::endl;
}
std::cout << " Came out of the loop" << std::endl;
std::cout << " x = " << (int) x <<  std::endl;
```

Code 5.0

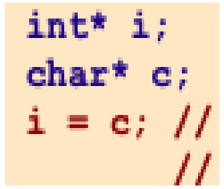
```cpp
int n0 = 7;
int n1 = 7.2;
int n2 {6};
int n3 = {5.5}; // = is redundant

std::cout << "n0 = " << n0 << std::endl;
std::cout << "n1 = " << n1 << std::endl;
std::cout << "n2 = " << n2 << std::endl;
```

Code 6.0



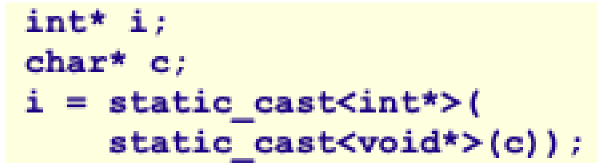
```cpp
int a[] {1, 2, 3, 4, 5, 6};
const auto n = 6;

for (auto i = 0; i < n; i++)
    std::cout << a[i] << ' ';
std::cout << std::endl;
```

Code 7.0 & Code 8.0



```
int* i;
char* c;
i = c; //
       //
```

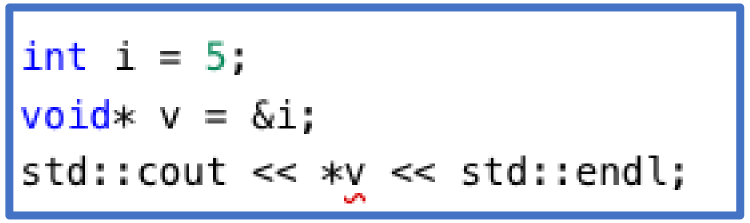
Code 7.0



```
int* i;
char* c;
i = static_cast<int*>(
    static_cast<void*>(c));
```

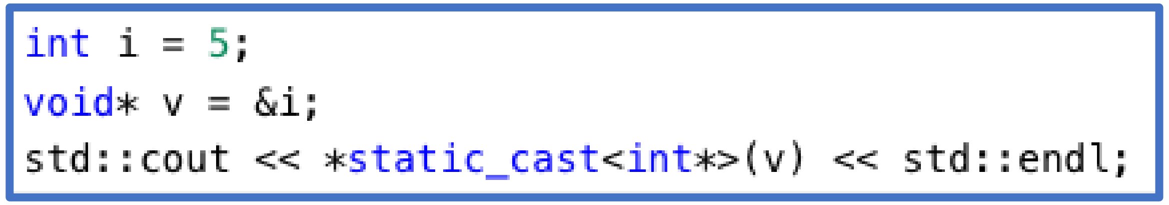
Code 8.0

Code 9.0 & Code 10.0



```
int i = 5;
void* v = &i;
std::cout << *v << std::endl;
```

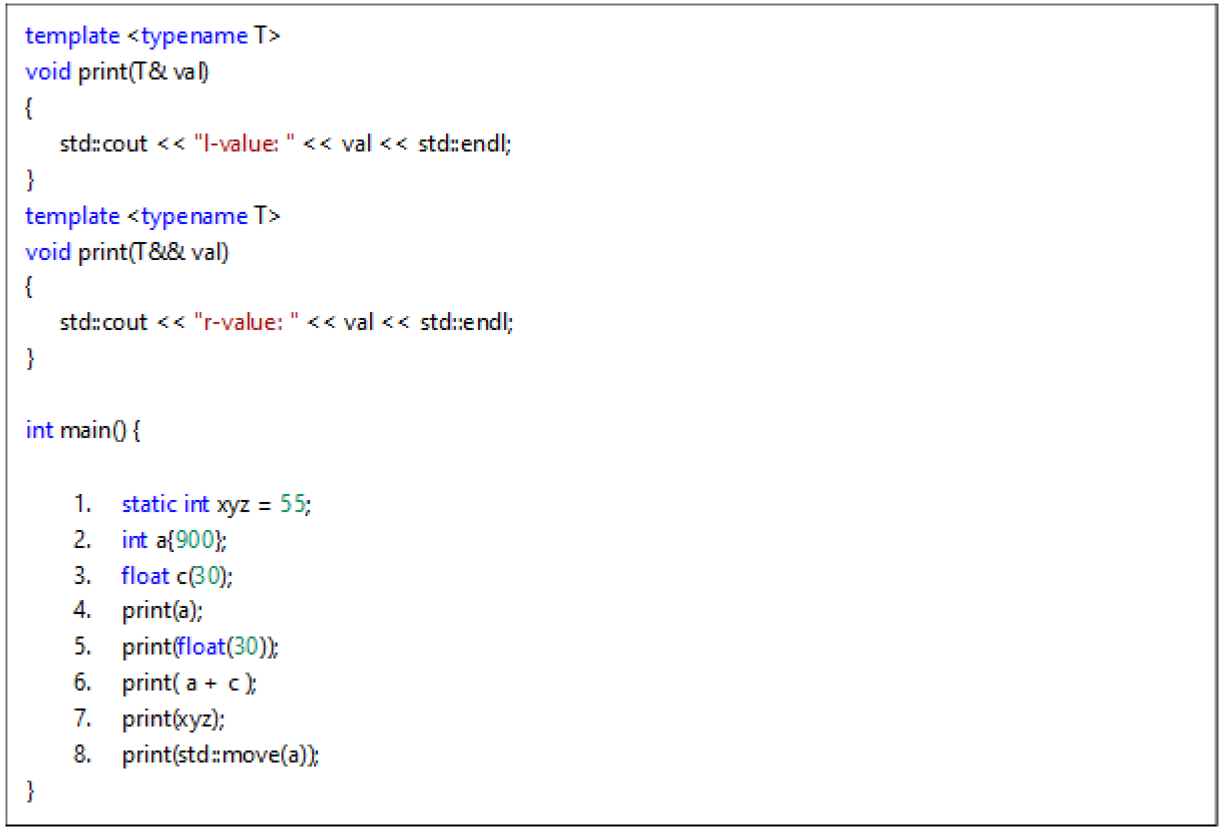
Code 9.0



```
int i = 5;
void* v = &i;
std::cout << *static_cast<int*>(v) << std::endl;
```

Code 10.0

Code 11.0



```
template <typename T>
void print(T& val)
{
    std::cout << "l-value: " << val << std::endl;
}
template <typename T>
void print(T&& val)
{
    std::cout << "r-value: " << val << std::endl;
}

int main() {

    1.  static int xyz = 55;
    2.  int a{900};
    3.  float c(30);
    4.  print(a);
    5.  print(float(30));
    6.  print( a + c );
    7.  print(xyz);
    8.  print(std::move(a));
}
```

Code 12.0

```
int foo (10);
auto bar = std::ref(foo);
++bar;
++foo
std::cout << foo << '\n';
```

Code 13.0

```cpp
int foo (10);
int bar;
bar = std::ref(foo);
++bar;
std::cout << foo << '\n';
std::cout << bar << '\n';
```

Code 14.0

```
int a[]{1, 2, 3, 4, 5, 6};
for (auto e : a){
  e += 2;
}
for (auto& e : a){
  e++;
}
for (auto& e : a){
  std::cout << e << ' ';
}
std::cout << std::endl;
```

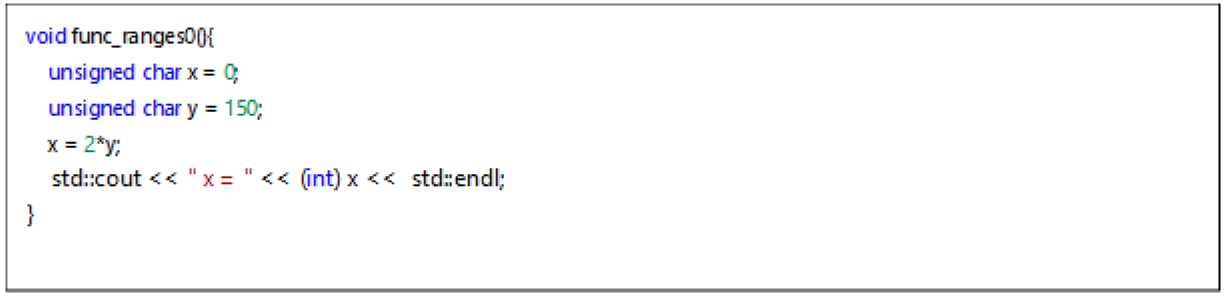
Code 15.0

```
class Subject {
     unsigned number;
     char desc[41];
     Subject preRequisite;
 };
```

Code 16.0

```
class Subject{
    const int id = 100;
    Subject(): id(5){
        id = 5;
    }
};
```

Code 17.0



```
void func_ranges0(){
   unsigned char x = 0;
   unsigned char y = 150;
   x = 2*y;
    std::cout << " x = " << (int) x <<  std::endl;
}
```