Advanced Programming

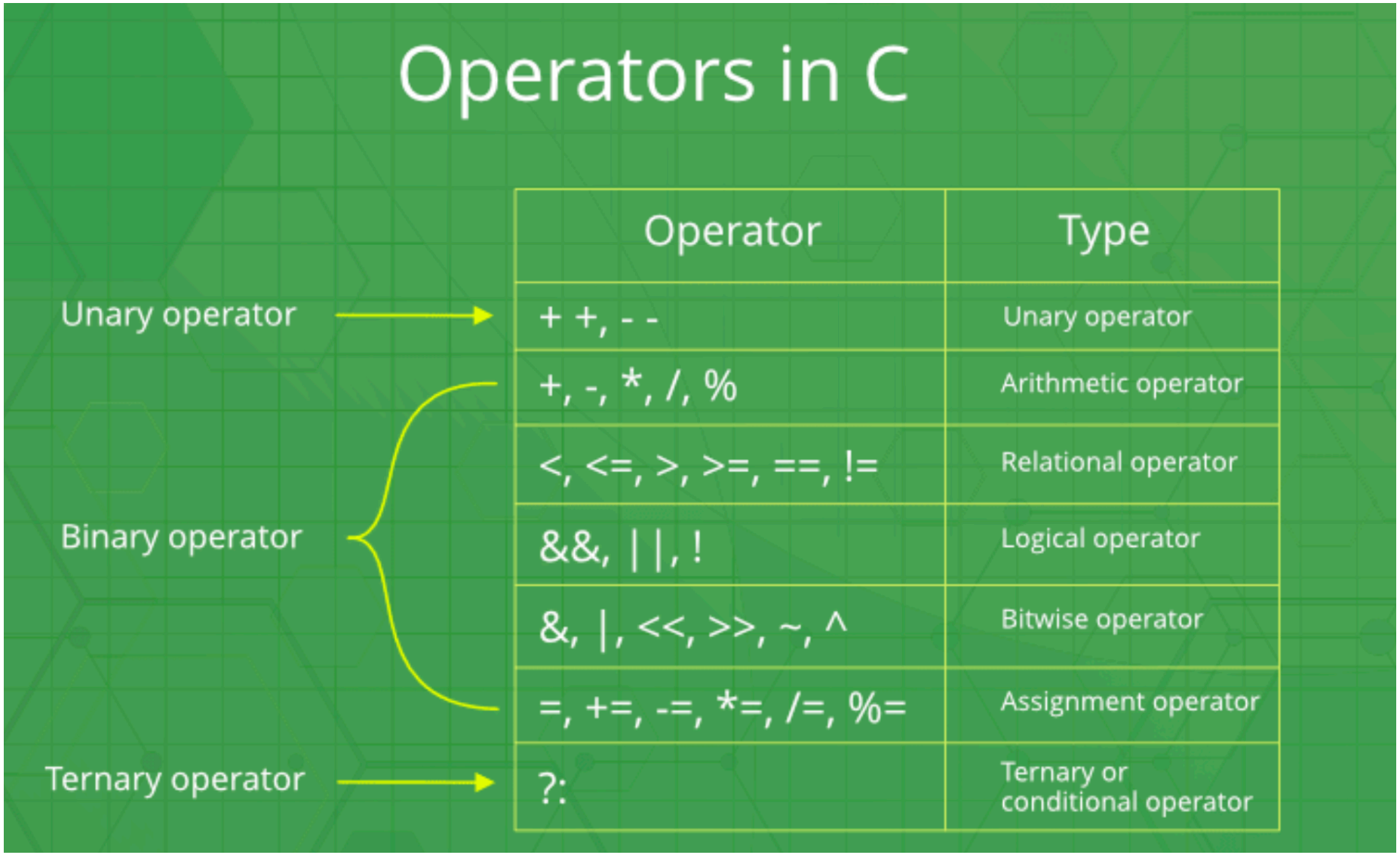
ACSE-5: Lecture 4

Adriana Paluszny

Overview

- STL Containers: iterators and algorithms
- Debug
- Object oriented programming:
  - copy constructor, members, operators
  - Encapsulation: mutators and accessors
- Inheritance
- Polymorphism

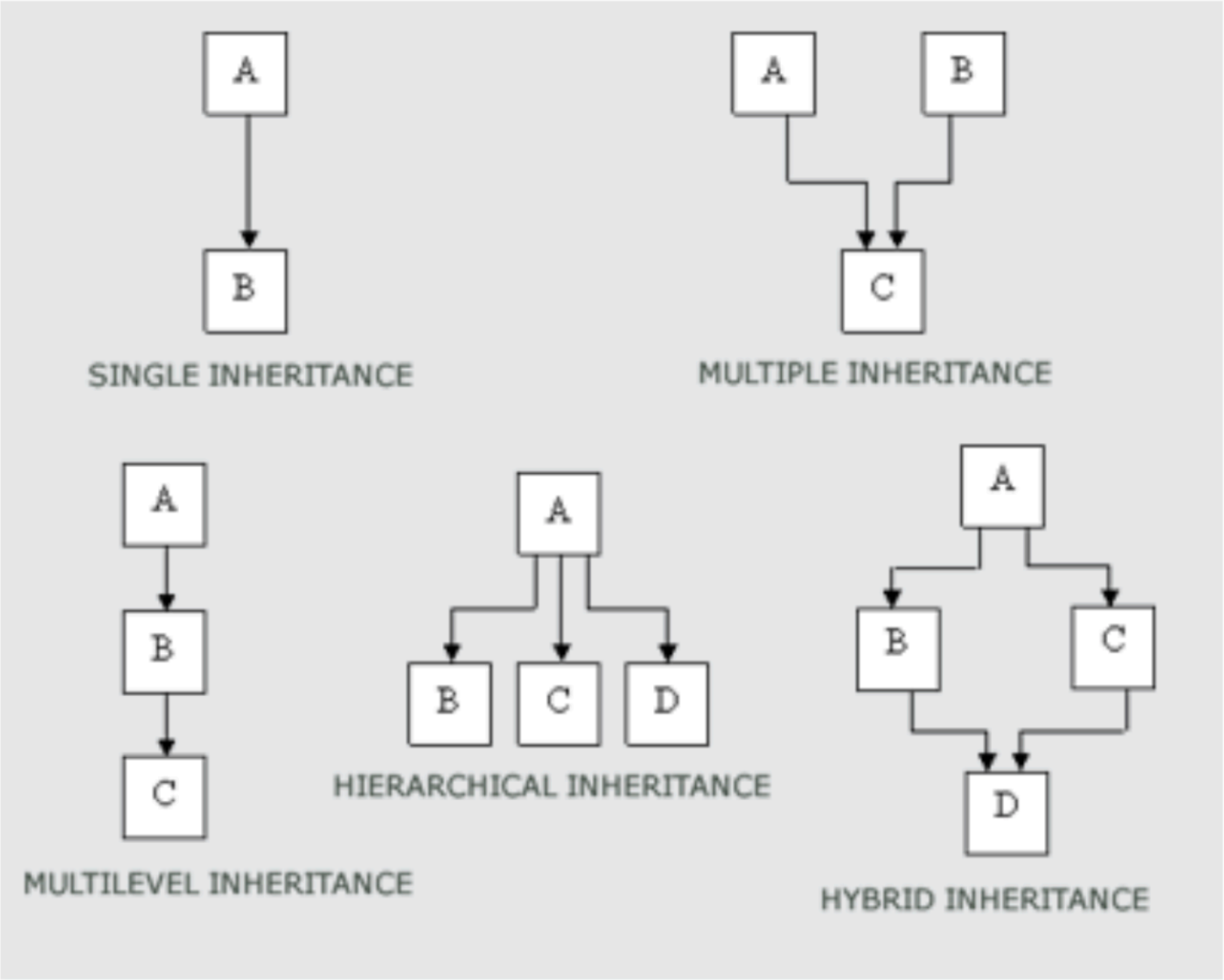
Operators


Operators in C
Operator
Type
Unary operator
+ +, - -
Unary operator
+, -, *, /, %
Arithmetic operator
<, <=, >, >=, ==, !=
Relational operator
Binary operator
&&, ||, !
Logical operator
&, |, <<, >>, ~, ^
Bitwise operator
=, +=, -=, *=, /=, %=
Assignment operator
Ternary operator
?:
Ternary or conditional operator

Types of inheritance

Inheritance can also be:
Private
Protected
Public

Both members and functions can be inherited



Homework

1) **Read Chapters 7 and 8 of** Programming: Principles and Practice using C++ by Bjarne Stroustrup
2) Introduction to the C++ grammar: **chapter 6**
3) Google C++ exercises (next two slides)
4) Finish submission of assignment!

[from google c++ developers course]

- What's the output of the following program? Please do not run the program, but draw the memory picture to determine the output.
- Once you have determined the output by hand, run the program to see if you are correct.

```
void Unknown(int *p, int num);
void HardToFollow(int *p, int q, int *num);

void Unknown(int *p, int num) {
  int *q;

  q = &num;
  *p = *q + 2;
  num = 7;
}

void HardToFollow(int *p, int q, int *num) {
  *p = q + *num;
  *num = q;
  num = p;
  p = &q;
  Unknown(num, *p);
}

main() {
  int *q;
  int trouble[3];

  trouble[0] = 1;
  q = &trouble[1];
  *q = 2;
  trouble[2] = 3;

  HardToFollow(q, trouble[0], &trouble[2]);
  Unknown(&trouble[0], *q);

  cout << *q << " " << trouble[0] << " " << trouble[2];
}
```

[from google c++ developers course]

Consider this programme.

There is a line in this program marked "How does this line work?" - can you figure it out? Here is Google's explanation: https://developers.google.com/edu/c++/solutions/3-1

Write a program that initializes a 3-dim array and fills the 3rd dimension value with the sum of all three indexes. Here is Google's solution: https://developers.google.com/edu/c++/solutions/3-2

```
const int kStudents = 25;
const int kProblemSets = 10;

// This function returns the highest grade in the Problem Set array.
int get_high_grade(int *a, int cols, int row, int col) {
  int i, j;
  int highgrade = *a;

  for (i = 0; i < row; i++)
    for (j = 0; j < col; j++)
      if (*(a + i * cols + j) > highgrade)  // How does this line work?
        highgrade = *(a + i*cols + j);
  return highgrade;
}

int main() {
 int grades[kStudents][kProblemSets] = {
   {75, 70, 85, 72, 84},
   {85, 92, 93, 96, 86},
   {95, 90, 83, 76, 97},
   {65, 62, 73, 84, 73}
 };
 int std_num = 4;
 int ps_num = 5;
 int highest;

 highest = get_high_grade((int *)grades, kProblemSets, std_num, ps_num);
 cout << "The highest problem set score in the class is " << highest << endl;

 return 0;
}
```