CS2313 Computer Programming

LT10 – Class

In-class Questions

Question 1:

What is an object in C++?

A. A function that performs a specific task.
B. A variable of a primitive data type.
C. An instance of a class.
D. A keyword in C++.

Question 2:

In C++, what is the purpose of the private access specifier in a class?

A. To allow access to all functions and objects.
B. To restrict access to only member functions of the same class.
C. To make the members accessible from anywhere in the program.
D. None of them above.

Overloading operators

- Operators can be overloaded in 2 ways:

  As a friend function

  As a member function

Questions for Group Discussion

- What is the output of following code?

```
class complex {
   int i;
   int j;
public:
   complex(){}
   complex(int a, int b) { i = a; j = b; }
   complex operator+(complex c) {
          complex temp;
          temp.i = i + c.i;
          temp.j = j + c.j;
          return temp; }
   void show(){
          cout<<"Complex Number:
};

int main(){
   complex c1(1,2);
   complex c2(3,4);
   complex c3 = c1 + c2;
   c3.show();
   return 0;
}
```

Overloading an operator using member function:

1. The overloaded operator must be added as a member function of the left operand.
2. All other operands become function parameters.

Questions for Group Discussion

- What is the output of following code?

```
class complex {
   int i;
   int j;
public:
   complex(){}
   complex(int a, int b) { i = a; j = b; }
   complex operator+(complex c) {
          complex temp;
          temp.i = i + c.i;
          temp.j = j + c.j;
          return temp; }
   void show(){
          cout<<"Complex Number:
};

int main(){
   complex c1(1,2);
   complex c2(3,4);
   complex c3 = c1 + c2;
   c3.show();
   return 0;
}
```

Overloading an operator using member function:

1. The overloaded operator must be added as a member function of the left operand.
2. All other operands become function parameters.

CS2313 Computer Programming

Game Design

Rock paper scissors

Outline

- Random Number

- Game Design: Rock Paper Scissors
  - Basic Level
  - Advance Level

Random Number

- Computers are “deterministic” machines
- Computers are not capable of true randomness
- Instead, pseudo-random numbers are used
  - Pseudo-random numbers are generated using a mathematical formula
  - There is a pattern in pseudo-random numbers
  - The pattern is nearly impossible to observe

Random Number

- Functions provided:
  - void srand(unsigned int seed);
    - This function sets the seed for the random number generator
  - int rand();
    - This function generates and returns an integer value in the range 0..RAND_MAX

Random Number

- srand() is usually only called ONE time
- rand() is called every time a random number is desired
  - If you want a number between 0 and N
    - val = rand() % (N+1)
  - If you want a number with a range M..N
    - val = rand() % (N-M+1) + M

[0,N-M] [M,N]

In-class Questions

Question 1:

If you want to generate a random number between 0 and N, which expression should you use?

A. rand() % N

B. rand() % (N + 1)

C. rand() % (N - 1)

D. rand() + N

Question 2:

What does rand() % (N - M + 1) + M do?

A. Generates a random number between 0 and N.

B. Generates a random number between M and N.

C. Generates a random number between N and M + N.

D. Generates a random number between M - 1 and N + 1.

In-class Questions

Question 3:

Which of the following expressions generates a random number between -10 and 10?

A. rand() % 21 - 10
B. rand() % 20 + 10
C. rand() % (10 + 1)
D. rand() % 21 + (-10)

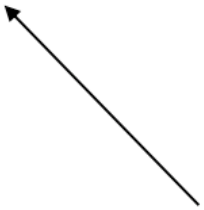
Question 4:

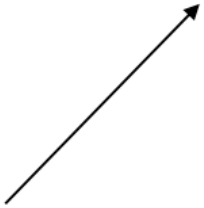
To simulate a dice roll with values between 1 and 6, which expression should you use?

A. rand() % 5 + 1
B. rand() % 6 + 1
C. rand() % 7
D. rand() % 6

Random Number

Enter seed: 12	Enter seed: 1652	Enter seed: 12
42	38	42
46	32	46
40	43	40
41	43	41
40	36	40
43	48	43
32	34	32
38	48	38
31	31	31
42	49	42





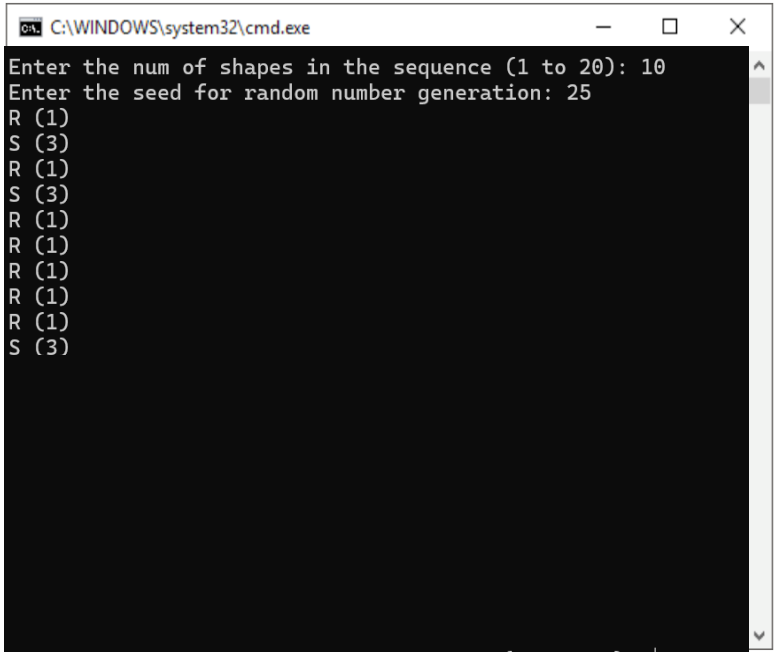
Note: Same seed = same sequence = same results

Rock-Paper-Scissors Game

- Basic Level
  - To Initialize a random sequence of Shapes
- Advanced Level
  - To play the Rock-Paper-Scissors game
- Shapes
  - Rock, Paper, or Scissors

Basic Level

- Expected outcomes

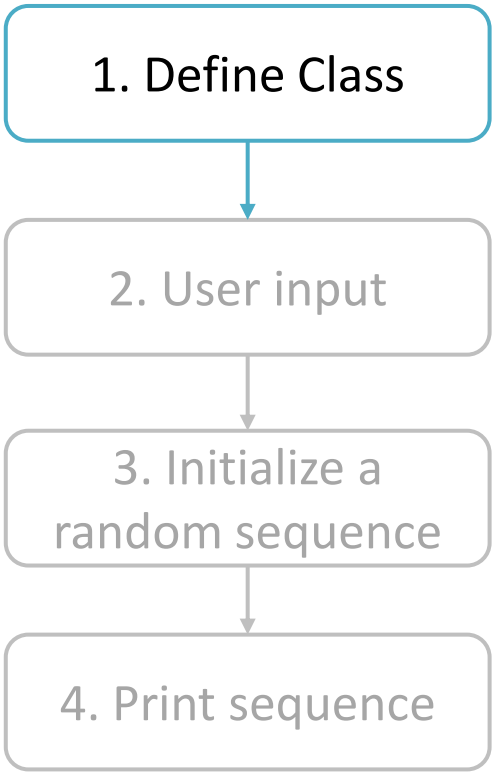


- **User Inputs**:
  - The number of rounds
  - A seed value for the random number generator

- **Output**:
  - A random sequence

Basic Level

- Program logic

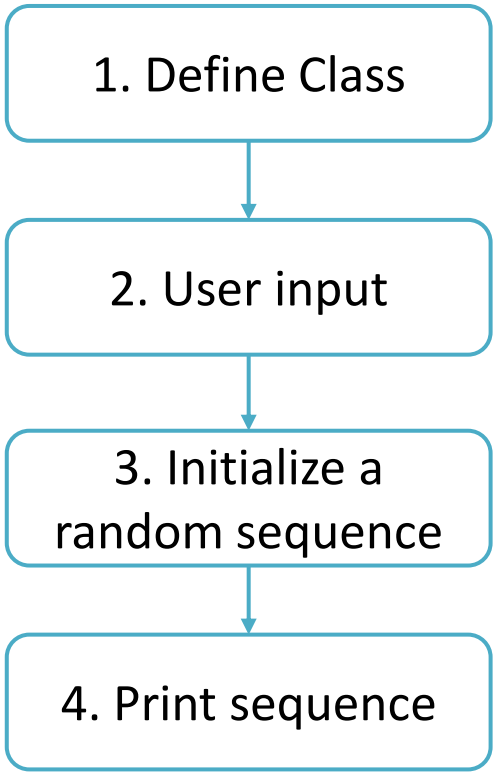



```
class Shape
{
public:
  Shape(); // default constructor
  void setName(char n); // set the name of the shape
  char getName(); // get the name of the shape
  void setValue(int v); // set the corresponding value of the shape
  int getValue(); // get the corresponding value of the shape


private:
  char name; // name of the shape, S(Scissors), R(Rock), P(Paper)
  int value; // corresponding value of the shape, 1(Rock), 2(Paper), 3(Scissors)
};
```

Basic Level

- Program logic



1

2

3

4

```
// Declare the functions
void initSequence(Shape shapeSeq[], char shapeName[], int num);
void printSequence(Shape shapeSeq[], int num);
void determineWinner(Shape shape_user, Shape shape_computer, int round);

// maximum number of shapes in the sequence
const int MAX_NUM = 20;

int main()
{
 Shape shapeSeq[MAX_NUM];  // sequence of shapes


 char shapeName[3] = { 'R', 'P', 'S' };  // name of the shapes, R(Rock), P(Paper), S(Scissors)


 int num = 0;  // number of shapes in the sequence
 cout << "Enter the num of shapes in the sequence (1 to 20): ";
 cin >> num;


 int seed;  // seed for random number generation
 cout << "Enter the seed for random number generation: ";
 cin >> seed;
 srand(seed);    // set the seed for random number generation


 initSequence(shapeSeq, shapeName, num);
 printSequence(shapeSeq, num);

 return 0;
 // play the game
}
```

Basic Level

- Class
  - *Shape*
    - A specific move in the game
    - Member function/variable of a *Shape*
      - *Name*
      - *Value*

	Rock	Paper	Scissors
Name	**R**	**P**	**S**
Value	1	2	3

Basic Level

- Function design
  - initSequence(int num)
    - Initialize the sequence of shapes
    - Randomly select the shape value from 1 to 3
    - The integer value *num* indicates the number of shapes in this sequence
  - printSequence()
    - Print each *Shape* object sequentially.

In-class Questions

Question 1:

What should the *initSequence* function do with the num parameter?

A. num represents the maximum possible value for Shape values, so it should be used in rand() % num.

B. num represents the number of Shape objects in shapeSeq to initialize.

C. num is the index of the last initialized Shape.

D. num is not needed in initSequence.

Question 2:

In *initSequence*, the goal is to initialize each element of *shapeSeq* with a name and a random value. Arrange the following statements in the correct order for this purpose:

1.shapeSeq[i].name = shapeName[shapeSeq[i].value];

2.shapeSeq[i].value = rand() % 3;

3.for (int i = 0; i < num; i++)

A. 3, 1, 2

B. 1, 3, 2

C. 3, 2, 1

D. 2, 3, 1

In-class Questions

Question 3:

Which of the following statements is likely to be used in *printSequence* to print each *Shape* object in *shapeSeq*?

A. cout << shapeSeq[i].name << " (" << shapeSeq[i].value << ") " << endl;

B. cout << shapeSeq.name << " (" << shapeSeq.value << ") " << endl;

C. cout << shapeSeq.name << shapeSeq[i].value;

D. cout << shapeSeq[i].value << shapeSeq.name;

Question 4:

Assume num represents the number of *Shape* objects in *shapeSeq*. Which of the following correctly describes the loop structure in *printSequence* to print each Shape object?

A. for (int i = 1; i <= num; i++)

B. for (int i = 0; i <= num; i++)

C. for (int i = 0; i < num; i++)

D. for (int i = num; i >= 0; i--)

Basic Level

Function design

Initialize the sequence of shapes

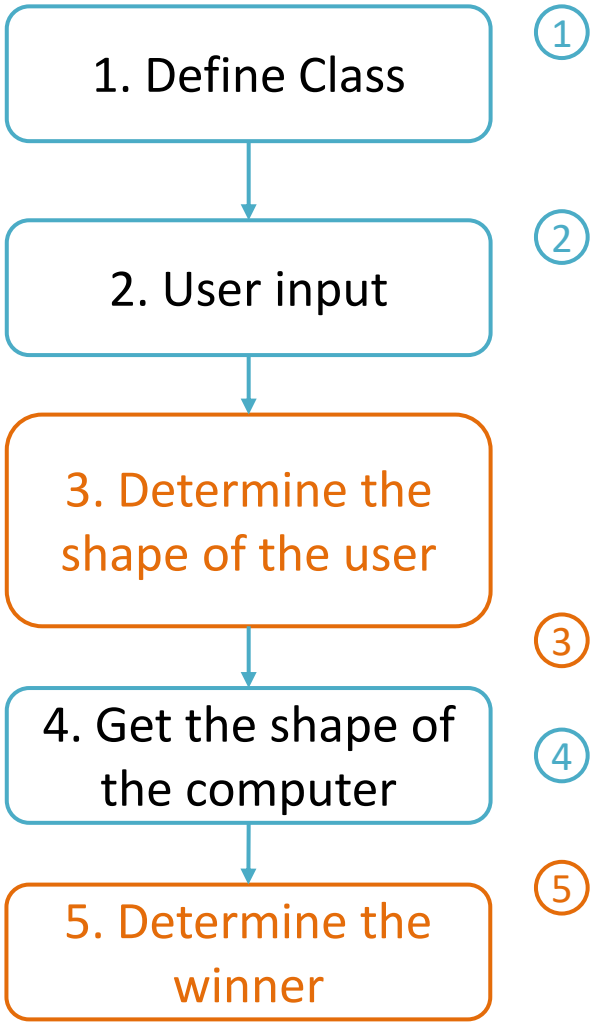
```
void initSequence(Shape shapeSeq[], char shapeName[], int num)
{
  for (int i = 0; i < num; i++)  // iterate through the sequence of shapes
  {
    int j = rand() % 3;  // generate a random number in {0, 1, 2}
    shapeSeq[i].setName(shapeName[j]);  // set the name of the shape (Note: why start from 0?)
    shapeSeq[i].setValue(j + 1);  // set the value of the shape
  }
}
```

Print each *Shape* object sequentially

```
void printSequence(Shape shapeSeq[], int num)
{
  for (int i = 0; i < num; i++)
  {
    cout << shapeSeq[i].getName() << " (" << shapeSeq[i].getValue() << ")" << endl;
  }
}
```

Play the Game!

- Program logic




```
int main()
{
 Shape shapeSeq[MAX_NUM];  // sequence of shapes

 char shapeName[3] = { 'R', 'P', 'S' };  // name of the shapes, R(Rock), P(Paper), S(Scissors)


 // User input: number of shapes in the sequence
 // User input: seed for random number generation

 // play the game
 for (int i = 0; i < num_game; i++)
 {
  int shape_value_user = 0;  // shape selected by the user
  cout << "Enter your shape (1 for Rock, 2 for Paper, 3 for Scissors): ";
  cin >> shape_value_user;
  Shape shape_user;


  // determine the shape selected by the user


  // get the shape selected by the computer
  Shape shape_computer = shapeSeq_computer[i];


  // determine the winner of the round and print the result
  determineWinner(shape_user, shape_computer, i);
 }
 return 0;
}
```

3.Determine the shape of the user

```
// play the game in main ()
for (int i = 0; i < num_game; i++)
 {
   int shape_value_user = 0;  // shape selected by the user
   cout << "Enter your shape (1 for Rock, 2 for Paper, 3 for
Scissors): ";
   cin >> shape_value_user;
   Shape shape_user;

   // determine the shape selected by the user
   if (shape_value_user == 1)  // Rock
   {


   }
   else if (shape_value_user == 2)  // Paper
   {


   }
   else if (shape_value_user == 3)  // Scissors
   {


   }
```

```
else  // deal with invalid input
   {
     cout << "Invalid input. Please enter 1, 2, or 3." << endl;
     i--;
     continue;
   }
   // get the shape selected by the computer
   Shape shape_computer = shapeSeq_computer[i];
   // determine the winner of the round and print the result
   determineWinner(shape_user, shape_computer, i);
 }
```

Given *Shape* class

```
class Shape
{
public:
  Shape();
  void setName(char n);
  char getName();
  void setValue(int v);
  int getValue();

private:
  char name;
  int value;
};
```

5. Determine the winner

```
// determines the winner of the round
void determineWinner(Shape shape_user, Shape shape_computer, int round)
{    /*============================================================
      * Complete the code which determines the winner of the round
      * The rules are as follows:
      * Rock (1) beats Scissors (3)
      * Scissors (3) beats Paper (2)
      * Paper (2) beats Rock (1)
      * If the shapes are the same, it is a tie
      * The output should be in the format: Round i: User (shape_user) vs. Computer (shape_computer): Who wins
      * For example, if the user selects Rock, and the computer selects Paper, the output should be:
      * Round 1: User (R) vs. Computer (P): Computer wins
      * ============================================================*/
    if (                                                    )
    {
        cout << "Round " << round + 1 << ": User (" << shape_user.getName() << ") vs. Computer (" << shape_computer.getName() <<
"): Tie" << endl;
    }
    else if (
                                                                                )
    {
        cout << "Round " << round + 1 << ": User (" << shape_user.getName() << ") vs. Computer (" << shape_computer.getName() <<
"): User wins" << endl;
    }
    else
    {
        cout << "Round " << round + 1 << ": User (" << shape_user.getName() << ") vs. Computer (" << shape_computer.getName() <<
"): Computer wins" << endl;
    }
}
```

Advance Level

- Description
  - Initially, you have $10 dollars, you compete with computer.
    - If you win, your balance will increase $10 dollars.
    - If you lose, it will decrease $10 dollars.
    - If draw game, your balance will not be changed.
  - The game continues until one of these conditions are met:
    - Your balance becomes 0.
    - You choose to leave the game with a positive balance.
    - The number of rounds has reached 10