CS 31
Discussion 2H

Switch Statement

Switch

- Syntax:

```
switch (expression)
{
    case constant1:
        group_of_statements_1;
        break;
    case constant2:
        group_of_statements_2;
        break;
    ...
    default:
        default_group_of_statements
}
```

- Rules:
- Evaluates expression and checks if it is equivalent to constant1; if it is, executes group_of_statements_1, **until it finds the break statement**.
- When it finds the break statement, the program jumps to the end of the **entire** switch statement.
- If expression was not equal to constant1, check against constant2. If equal, execute group_of_statements_2, **until a break is found, when it jumps to the end of the switch**.
- Finally, if the value of expression did not match any of the constants (can be any number of these), the program executes the statements included after the "**default**" label, if it exists (since it is optional).

Switch

- Syntax:

```
switch (expression)
{
    case constant1:
        group_of_statements_1;
        break;
    case constant2:
        group_of_statements_2;
        break;
    ...
    default:
        default_group_of_statements
}
```

- Whenever meets a break, jump out of the entire switch statement; Or keep on executing.

- To check for a value among a number of possible **constant expressions**
- Similar to concatenating if-else statements, but **limited to constant expressions** (note that if.. else statements can be used to check not only constant expressions, but also ranges)
- If break is not included, all statements following the case (including those under any other labels) are **also** executed, until the end of the switch block or a jump statement (such as break) is reached.

- The following will act the same way:

```
switch (variable) {
    case value1:
        statement_1
        break;
    case value2:
        statement_2
    case value3:
        statement_3
        break;
    default:
        statement_4
        break;
```

```
if (variable == value1)
    statement_1
else if (variable == value2)
    statement_2
    statement_3
else if (variable == value3)
    statement_3
else
    statement_4
```

Switch and If

- The following will act the same way:

```
switch (x) {
    case 1:
        cout << "x is 1";
        break;
    case 2:
        cout << "x is 2";
        break;
    default:
        cout << "value of x unknown";
    }
```

```
if (x == 1) {
    cout << "x is 1";
}
else if (x == 2) {
    cout << "x is 2";
}
else {
    cout << "value of x unknown";
}
```

Functions

- Functions are sections of code that do a particular task.
- Break large problem into smaller sub problems
- Each sub problem is expressed as a function call
- Similar to what we have in mathematics
  - `f(x) = 0.5 * x * (x+1)`

Functions

```
return_type function_name
(parameter1_type parameter1_name, parameter2_type parameter2_name, ...)
{
    // function body
}
```

- **Parameters** are the ZERO or MORE inputs to your functions. Each has its type defined, and a placeholder name assigned to it.
- **Return type** is the type of value I expect to receive back from a function (e.g. ints, doubles, strings, etc.)
- **Arguments** are what you call the values you pass into a function when you're **calling** it.

Functions

```
double f(int x)
{
    double solution = 0.5 * x * (x + 1);
    return solution;
}
```

f: Name of the function

x: Input parameter of type ‘int’

Return type is ‘double’

{....} is the body of the function

Parameter Passing

- Pass-By-Value Scheme Is What We Have Seen So Far
  - Functions See A Copy Of The Value Passed, Not The Value Itself
  - *i*-th Formal Parameter Is A Local Variable Initialized To The *i*-th Actual Argument
- There Are Other Passing Schemes We'll Mention Later

Scope: Visibility of a Variable

- A variable is only visible in a particular block of code
  - **Global scope**: a variable declared outside any block is called global variable
    - Can be used anywhere in the program
    - Convention: Only define constant variables globally
  - **Block scope**: a variable declared within a block, such as a function or a selective statement, is called local variable
    - The visibility of a variable with block scope extends until the end of the block, **including** inner blocks.
    - An inner block can reuse a name existing in an outer scope to refer to a different variable (because it is a different block)
      - In this case, the name will refer to a different variable only within the inner block, hiding the variable which has the same name outside (Since they are different variables, what's happened inside will not happen outside to the variable with the same name).

Scope: Visibility of a Variable

- What's declared inside the braces is not visible outside the braces
  - Lifecycle: they are dead before go outside!!
- What's declared outside the braces is "global" to the statements inside the braces

```
#include <iostream>
using namespace std;

int main() {
    int i = 1, j = 2;
    cout << i + j << endl;
}
```

```
#include <iostream>
using namespace std;

int main() {
    { int i = 1, j = 2; }
    cout << i + j << endl;
}
```

i & j: they are dead outside the braces.

Scope: Visibility of a Variable

- What's declared inside the braces is not visible outside the braces
  - Lifecycle: they are dead before go outside!!
- What's declared outside the braces is "global" to the statements inside the braces (there's only one exception)

```
#include <iostream>
using namespace std;

int i = 1, j = 2;

int addition();
int main() {
    cout << addition() << endl;
}

int addition() {
    return (i+j);
}
```

```
#include <iostream>
using namespace std;

int addition();
int main() {
    int i = 1, j = 2;
    cout << addition() << endl;
}

int addition() {
    return (i+j);
}
```

i & j: they are dead outside the `main()` function.

Time for practicing the Worksheet

Problem #1: What is the output of the program below?

```
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

int main()
{
        const string str = "There is an Apple!";
        int count=0;

        for(int i=0 ; i != str.size() ; i++)
                if( str[i] == 'a' && isalpha(str[i]) )
                        count++;

        cout << count << endl;
}
```

(1) 1
(2) 2
(3) 3
(4) 4
(5) 5
(6) No output. There is a Compilation Error.

Problem #2: Which of the following code segments produce the exact output as the sample program provided below? Suppose each choice uses the same header files (#include…).

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        for(int i=0;i<2;i++) {
                for(int j=0;j<=i;j++)
                        cout << "*" ;
                cout << endl;
        }
}
```

(1)

```
int main()
{
        for(int i = 2 ; i >= 1 ; i-- ) {
                for(int j = 0 ; j <= i ; j++ )
                        cout << "*" ;
                cout << endl;
        }
}
```

(2)

```
int main()
{
        for(int i = 2 ; i >= 1 ; i-- ) {
                for(int j = 0 ; j < i ; j++ )
                        cout << "*" ;
                cout << endl;
        }
}
```

(3)

```
int main()
{
        for(int i = 2 ; i >= 1 ; i-- ) {
                for(int j = 3 ; j > i ; j-- )
                        cout << "*" ;
                cout << endl;
        }
}
```

(4)

```
int main()
{
        for(int i = 2 ; i >= 1 ; i-- ) {
                for(int j = 2 ; j >= i ; j-- )
                        cout << "*" ;
                cout << endl;
        }
}
```

(5)

```
int main()
{
        for(int i = 2 ; i >= 1 ; i-- ) {
                for(int j = i+1 ; j >= i ; j-- )
                        cout << "*" ;
                cout << endl;
        }
}
```

(6)

```
int main()
{
        for(int i = 5 ; i >= 4 ; i-- ) {
                for(int j = 5 ; j >= i ; j-- )
                        cout << "*" ;
                cout << endl;
        }
}
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        string str = "abcdefga";
        int value = 0;

        for(int i = 0 ; i != str.size() ; i++ ) {
                for(int j = i + 1 ; j != str.size() ; j++ ) {
                        if( str[ i ] == str[ j ] ) {
                                value = j - i;
                        }
                }
        }

        cout << value << endl;
}
```

(1) 1
(2) 2
(3) 3
(4) 4
(5) 5
(6) 6
(7) 7
(8) 8
(9) 9
(10) 10
(11) 11
(12) 12
(13) No output. There is a Compilation Error.

```
int main()
{
        int i = 0,value = 0;

        for( i = 0 ; i < 5 ; i++ ){
                switch( i ) {
                case 1: value = i + 1;
                case 2: value = value - 1;
                case 3: value = value + 1;
                case 4: value = value + 2; break;
                case 5: value = value - 1;
                default: value = value + 1;
                }
        }

        cout << value << endl;
}
```

(1) 1
(2) 2
(3) 3
(4) 4
(5) 5
(6) 6
(7) 7
(8) 8
(9) 9
(10) 10
(11) 11
(12) 12
(13) 13
(14) 14
(15) No output. There is Compilation Error

Problem #5: What is the output of the program below?

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
        int i = 0;

        do {
                for(int j = i ; j >= 0 ; j-- )
                        cout << "*" ;
                cout << endl;
                i++;
        }while(i<4);
}
```

(1)

```
*
**
***
****
```

(2)

```
*
**
***
```

(3)

```
***
**
*
```

(4)

```
****
***
**
*
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
	string exp = "-3+3+555+99999";

	int mult = 1;
	int offs = 0;
	for(int i=0;i != exp.size() ; i += 2) {
		if( exp[ i ] == '-' )
			mult = i+5;
		else if( exp[ i ] == '+' )
			mult = i+8;
		else
			offs = 1;
	}
	cout << exp[offs] << exp[mult % exp.size() ] << endl;
}
```

(1) 5+
(2) 99
(3) -3
(4) 3+
(5) 33
(6) 55
(7) 5+
(8) +5
(9) 35
(10) 53
(11) 59
(12) 39
(13) 93
(14) -9
(15) 9+

Problem #7: Please use switch-case to write a program below to output a student's grade based on the following conditions:
If a student's grade is from 90 to 100, then it's A (just cout << "A" << endl;)
If a student's grade is less than 90, but greater than or equal to 80, then it's B.
If a student's grade is less than 80, but greater than or equal to 70, then it's C.
If a student's grade is less than 70, but greater than or equal to 60, then it's D.
Otherwise, it's F.
If you do not use switch-case to write your program, you'll get zero points.
You can only use 1 extra variable. Failure to comply gets zero points.

```
#include <iostream>
using namespace std;

int main() {
    int score = 0;

    cout << "Please enter the student's score:";
    cin >> score;
```

```
int main() {
    int score = 0;

    cout << "Please enter the student's score:";
    cin >> score;

    int choice = score/10;

    switch(choice) {
        case 10:
        case 9:
            cout << "A" << endl;
            break;
        case 8:
            cout << "B" << endl;
            break;
        case 7:
            cout << "C" << endl;
            break;
        case 6:
            cout << "D" << endl;
            break;
        default:
            cout << "F" << endl;
    }
}
```

Problem #8: Please write a do-while loop to find the first lower case character in a given string. If there is no lower case letter, the program outputs "No lowercase character". If there is a lower case character, output the first lower case character. If your program gets stuck in an infinite loop, you'll get zero points.

```
#include <iostream>
#include <cctype>
#include <string>
using namespace std;

int main() {
        int i;

        string str;

        cin >> str;



   if (                                    )
           cout << "No lowercase character" << endl;
   else
           cout << str[i] << endl;
}
```

```
int main() {
        int i;

        string str;

        cin >> str;

        i = -1;

        do {
                i++;
        }while( i != str.size() && !islower(str[i]) );


        if (        i == str.size()        )
                cout << "No lowercase character" << endl;
        else
                cout << str[i] << endl;
}
```

2) What does the following code snippet output?

```
void mystery(char code) {
  switch(code) {
    case 'a':
    case 'b':
    case 'c':
      cout << "spooky";
      break;
    case 'd':
      cout << "feeling";
      break;
    case '1':
      cout << " ";
      break;
    case '2':
      cout << "?";
    default:
      cout << endl;
      break;
  }
}
```

```
int main() {
  string message = "d1a2c1d#";
  int i = 0;
  do {
    mystery(message[i]);
    i++;
  } while(i < message.length());
}
```

2) What does the following code snippet output?

```
void mystery(char code) {
  switch(code) {
    case 'a':
    case 'b':
    case 'c':
      cout << "spooky";
      break;
    case 'd':
      cout << "feeling";
      break;
    case '1':
      cout << " ";
      break;
    case '2':
      cout << "?";
    default:
      cout << endl;
      break;
  }
}
```

```
int main() {
  string message = "d1a2c1d#";
  int i = 0;
  do {
    mystery(message[i]);
    i++;
  } while(i < message.length());
}
```

feeling spooky?
spooky feeling

2) a) Write a function *isPalindrome* that takes in a string and determines if it is a palindrome. A palindrome is a string that reads the same forwards and backwards.

For example:
isPalindrome("abcba") returns true
isPalindrome("skt_will_win") returns false
isPalindrome("z") returns true
isPalindrome("") returns true

```
bool isPalindrome(string s) {
  int i = 0;
  int j = s.size() - 1;

  while (i < j) {
    if (s[i] != s[j]) {
      return false;
    }
    i++;
    j--;
  }
  return true;
}
```

b). Now write a function, *isPalindrome2*, that is similar to the function above, except we don't care about spaces in the string.

For example:
isPalindrome2("ggnore ero n g g") returns true

```
bool isPalindrome2(string s) {
  int i = 0;
  int j = s.size() - 1;

  while (s[i] == ' ') {
    i++;
  }
  while (s[j] == ' ') {
    j--;
  }

  while (i < j) {
    if (s[i] != s[j])
      return false;
    do {
      i++;
    } while (s[i] == ' ');
    do {
      j--;
    } while (s[j] == ' ');
  }
  return true;
}
```

4) Write a function *integerDivide* that does integer division without using the division operator (/). You can assume both parameters will be positive.

For example:

integerDivide(6, 2) returns 3

integerDivide(7, 2) returns 3

```
int integerDivide(int x, int y) {
   int count = 0;
   while (x >= y) {
     x -= y;
     count++;
   }
   return count;
}
```

5) Write a function *findLastLength* that takes in a string that consists of uppercase alphabetical characters, lowercase alphabetical characters, and empty space ' ' characters. It returns the length of the last word, unless the last word does not exist, in which case it returns 0.

For example:

findLastLength("Misfits should have won against SKT") returns 3

findLastLength(" ") returns 0

```
int findLastLength(string s) {
  bool foundWord = false;
  int lastLen = 0;
  for (int x = s.length() - 1; x >= 0; x--) {
    bool onAlpha = isalpha(s[x]);

    if (foundWord && !onAlpha)
      break;
    else if (onAlpha) {
      foundWord = true;
      lastLen++;
    }
  }
  return lastLen;
}
```

Writing PseudoCode

http://web.cs.ucla.edu/classes/spring18/cs31/

Thank You!