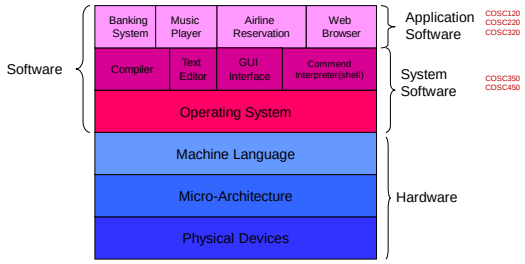
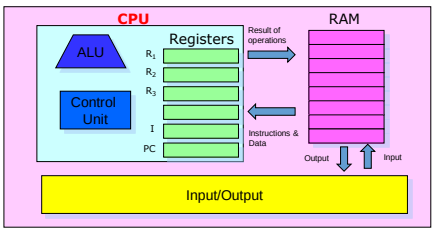
Review for COSC120

- Computer System
- Computer Structure
- C++ Environment
- Imperative vs. object-oriented programming in C++
- Input / Output
- Primitive data types
- Constant variable, static variable
- Formatted output
- Input from a file
- Output to a file

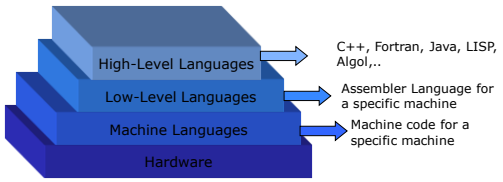
Review for COSC 120
Computer Systems



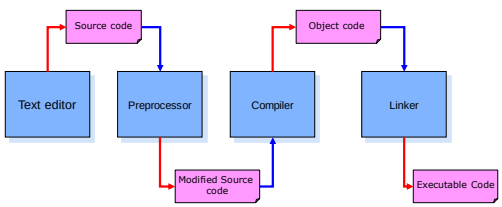
Review for COSC 120
Computer Structure (Von Newmann)



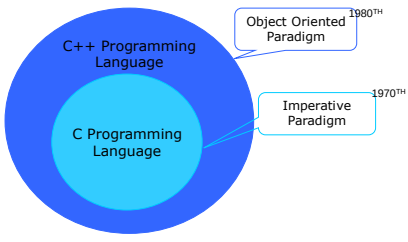
Review for COSC 120
High-level vs. Low-level Languages



Review for COSC 120
C++ System Environment



Review for COSC 120
Imperative vs. Object-Oriented Programming with C++



C++ is a super set of C

Review for COSC120
Input / Output

Review for COSC 120
Where we start to write a program?

1. Plan
   - Goals of program.
   - Inputs
   - Outputs
2. Modularization – Divide and Conquer
3. Write program module by module
4. compile it
5. Test the program

Review for COSC 120
Primitive Data Types(Integer)

Integer Types

short unsigned short	2 bytes 2 bytes	$-2^{16-1} \sim +2^{16-1}-1$ $0 \sim 2^{16}-1$
int unsigned int	4 bytes 4 bytes	$-2^{32-1} \sim +2^{32-1}-1$ $0 \sim 2^{32}-1$
long unsigned long	4 bytes 4 bytes	$-2^{32-1} \sim +2^{32-1}-1$ $0 \sim 2^{32}-1$

Review for COSC 120
Primitive Data Types (Decimal to Binary)

Ex) convert decimal number 13_{10} to binary

$13_{10} = 8 + 4 + 1 = 2^3 + 2^2 + 2^0$

$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 1101_2$

13 / 2 = 6, remainder 1

6 / 2 = 3, remainder 0

3 / 2 = 1, remainder 1

1 / 2 = 0, remainder 1

Review for COSC 120
Primitive Data Types (Binary to Decimal)

Ex) Convert a binary number 10001_2 to a decimal number

$10001_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 16 + 0 + 0 + 0 + 1$

$= 17_{10}$

Review for COSC 120
Integer representation in the Modern Computer System

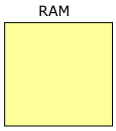
- Modern computer system use two's complementation for present a integer number
  - To represent a positive number, use the convention adopted for an unsigned integer.
  - To represent a negative number,
    1. Convert quantity of negative decimal to binary.
    2. Complement the positive number (change from 0 to 1 or from 1 to 0) – One's complement number.
    3. Add 1 to one's complemented number.
    4. Store in a memory

Review for COSC 120
Integer representation in the Modern Computer System

How to save integer + 25 in the 8 bit memory space

1. Convert 25 to binary number

00011001_2 2. Save in a memory



Review for COSC 120
Integer representation in the Modern Computer System

How to save negative integer –25 in the 8 bit memory space

1. Convert 25 to binary number

00011001

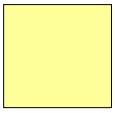
2. one's complementation

11100110

3. two's complementation

11100111

4. Save in a memory

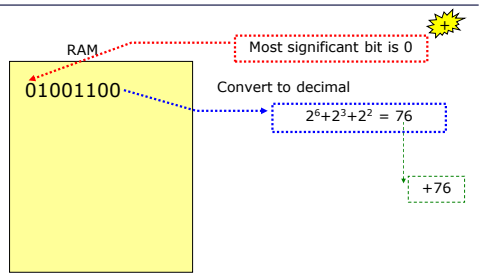


Review for COSC 120
Integer representation in the Modern Computer System

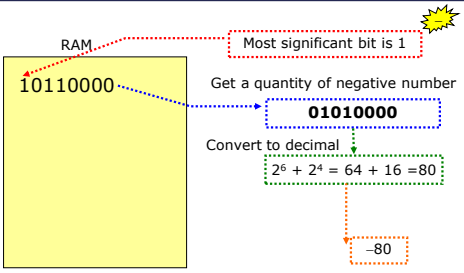
How to interpret a two's complement format integer value stored in the memory

- If the most significant bit is 0 (positive)
  - Change binary number to decimal
  - Put a plus sign in front of the number
- If the most significant bit is 1 (negative)
  - Leave the rightmost bits up to the first 1 unchanged. Complement the rest of the bits
  - Change the whole number from binary to decimal
  - Put a negative sign in front of the number.

Review for COSC 120
Integer representation in the Modern Computer System



Review for COSC 120
Integer representation in the Modern Computer System



Review for COSC 120
Primitive Data Types (Character)

- Each printable or non-printable character is represented by **one byte** unique number.
- The most commonly used method for encoding character is **ASCII** code (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange).

Review for COSC 120
Primitive Data Types (Character)

'A': a character **A** ASCII code 56. Need 1 byte to store

01000001

"A": a string **A** with length 1. Need 2 byte to store.

1 byte for ASCII code A (56) and 1 byte for the ASCII code for end of line character (\0).

01000001	00000000

Review for COSC 120
(Primitive Data Type Character: Coercion or Type Casting)

- C and C++ is one of the few languages to allow ***coercion***, that is forcing one variable of one type to be another type

```
// coercison.cpp demonstrates the coercion in C++
#include <iostream>
using namespace std;
int main()
{
        char letter;
        letter = 65;
        cout << letter << endl;
        letter = 66;
        cout << letter << endl;
        return 0;
}
```


Review for COSC 120
(Primitive Data Type Character: Coercion or Type Casting)

```
// coercion1.cpp demonstrates the coercion in C++

#include <iostream>
using namespace std;

int main()
{
  int letter;
  letter = 'A';
  cout << letter << endl;
  letter = 'B';
  cout << letter << endl;
  return 0;
}
```


Review for COSC 120
(Primitive Data Type Floating Point Number)

- Floating point data types are for real numbers.

float	4 bytes	$\pm 3.4 \times 10^{-38} \sim \pm 3.4 \times 10^{38}$
double	8 bytes	$\pm 1.7 \times 10^{-308} \sim \pm 1.7 \times 10^{308}$
long double	8 bytes	$\pm 1.7 \times 10^{-308} \sim \pm 1.7 \times 10^{308}$

- C++ support scientific notation for presenting a real number by using E notation

Decimal Notation	Scientific Notation	E Notation
123.45	**1.245×10^{2}**	**1.2345E2**
0.000012345	**1.245×10^{-5}**	**1.2345E-5**

Review for COSC 120
(Primitive Data Type Floating Point Number)

- When a floating-point value is assigned to an integer variable, the fraction part of value is discarded.

```
// float.cpp uses floating point data types.
#include <iostream>
using namespace std;
int main()
{
     int distance;

     distance = 7.89;
     cout << "The distance is " << distance << endl;
     return 0;
}
```


Review for COSC 120
(Primitive Data Type Boolean)

- Boolean variables only hold the values true or false.
- Usually false is assigned with 0 (zero) and true is assigned with positive integer.

```
// truefalse.cpp program uses Boolean data types.
#include <iostream>
using namespace std;
int main()
{
     bool TrueFalse;

     TrueFalse = true;
     cout << "True is print as " << TrueFalse << endl;
     TrueFalse = false;
     cout << "False is print as " << TrueFalse << endl;
     return 0;
}
```

Review for COSC 120
(Mathematical Expression)

- You can construct complex mathematical expression by using arithmetic operators and grouping symbol parenthesis.
- When you construct a mathematical expression, you need consider **associativity** and **precedence** of each operators used.

Review for COSC 120
(Constant Variable)

- A constant variable is a variable whose content cannot be changed
- A constant variable must be initialized when it is declared.

```
// constant.cpp demonstrate constant variable
#include <iostream>
using namespace std;
int main()
{
     const double PI = 3.14159; // declared and initiated
     double area, radius, circumference;

     cout << "This program calculates the area of a circle.\n";
     cout << "What is the radius of the circle? ";
     cin >> radius;
     area = PI * radius * radius;
     circumference = 2 * PI * radius;
     cout << "The area of circle is " << area << endl;
     cout << "The circumference of the circle is " << circumference <<endl;
     return 0;
}
```


Review for COSC 120
(The Static Variables)

Static Variable

- Defined locally
- One time initialization
- It only can be modified inside function, when the function is called
- It remains in memory until the end of the program

Review for COSC 120
(The Static Variables)

```
// staticx.cpp demonstrate local variable without static
#include <iostream>
using namespace std;

int function(int) //function prototype
int main()
{
    int z =0;
    for (int y=0; y<=5; y++)
    {
         z= function(10);
         cout <<"function returns "<< z <<endl;
    }
    return 0;
}

int function(int y)
{
    int x = 5;
    x = x + y;
    return x;
}
```


Review for COSC 120
(The Static Variables)

```
// static.cpp demonstrate local variable without static

#include <iostream>
using namespace std;

int function(int) //function prototype
int main()
{
    int z =0;
    for (int y=0; y<=5; y++)
    {
         z= function(10);
         cout <<"function returns "<< z <<endl;
    }
    return 0;
}
int function(int y)
{
    static int x = 5;
    x = x + y;
    return x;
}
```


Review for COSC 120
(Formatted Output with <iomanip> Library)

- **setw**(n)
- **setprecision**(n)
- **fixed** manipulator with **setprecision**(n)
- **showpoint** manipulator
- **left** and **right** manipulator

Review for COSC 120
(File Processing: Reading Data From a File)

Read data From a File

1. include the fstream header file;
2. declare a variable of type ifstream – create object to handle a file
3. open the file – by using the function open in ifstream object
4. check for an open file error
5. read data from the file
6. after each read, check for end-of-file using the eof() member function
7. close the file when access is no longer needed

Review for COSC 120
(File Processing: Reading Data From a File)

```
// readfile.cpp reads values from the file 'outfile.txt'
// and echoes them to the display until a negative value
// is read.
#include <iostream>
#include <fstream>
#include <cstdlib> // for exit function

using namespace std;

int main()
{
        ifstream indata; // create ifstream object to read a data
        int num;
        indata.open("infile.txt", ios::in); // opens the file
        if(!indata)  // checking whether 'outfile.txt' can open or not
        {
                cerr << "Error: file could not be opened" << endl;
                exit(1);
        }
        indata >> num;
        while (!indata.eof()) // keep reading until end-of-file
        {
                cout << "The next number is " << num << endl;
                indata >> num; // sets EOF flag if no value found
        }
        indata.close(); //close the file 'outfile.txt'
        cout << "End-of-file reached.." << endl;
        return 0;
}
```


Review for COSC 120
(File Processing: Writing Data To a File)

Writing Data To a File

1. include the **fstream** header file;
2. declare a variable of type **ofstream** – create **ofstream** object
3. open the file
4. check for an open file error
5. use the file
6. close the file when access is no longer needed

Review for COSC 120
(File Processing: Writing Data To a File)

```
// This program output values from an array
// to a file named outfile.txt
#include <iostream>
#include <fstream>
#include <cstdlib> // for checking error for opening a file
using namespace std;
int main()
{
        ofstream outdata;
        int i;
        int num[5] = {4, 3, 6, 7, 12};

        outdata.open("outfile.txt",ios::out);// create a file
        if( !outdata )
        {
                cerr << "Error: file could not be opened" << endl;
                exit(1);
        }
        for (i=0; i<5; ++i)
                outdata << num[i] << endl; // write data to the file
        outdata.close(); // close file
        return 0;
}
```


Review for COSC 120
(File Processing: Reading Data From a File more than one time)

- Read data more than one time from the beginning of a File
  - clear()
  - seekg(0)
- Sequence of two function calls let file pointer move to the beginning of a file!


```
// readagain.cpp reading a sequential access file example
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
using namespace std;
int main()
{
    ifstream sequential;
    sequential.open("input.txt",ios::in);
    if (!sequential)
    {
            cerr<<" File cannot be opened this time"<<endl;
            exit(1);
    }
    int age;
    char name[20];
    cout <<"     List of Man    "<<endl;
    cout <<"********************"<<endl;

    cout <<"Name"<<setw(20)<<"Age \n";
    cout <<"----"<<setw(20)<<"--- \n";
    while (sequential >> name >>age)
    {
            if (age > 18)
            cout <<name<<setw(20)<< left<<age <<setw(20)<<endl;
    }
    sequential.clear();
    sequential.seekg(0);
    cout <<endl<<endl;
    cout <<"  List of Children    "<<endl;
    cout <<"********************"<<endl;

    cout <<"Name"<<setw(20)<<right<<"Age \n";
    cout <<"----"<<setw(20)<<"--- \n";
    while (sequential >> name >>age)
    {
            if (age <= 18)
            cout <<left <<name<<setw(20)<< age<<setw(20)<<endl;
    }
    return 0;
```

Review for COSC 120

(Relational Operators)

- Relational operators are used to compare numerical values.
- A comparison result is true or false.

Relational Operators	
Operator Name	**Syntax**
Less Than	`a < b`
Less Than or Equal To	`a <= b`
Greater Than	`a > b`
Greater Than or Equal To	`a >= b`
Not Equal To	`a != b`
Equal To	`a == b`