**Templates**

**main.cpp**

#include <iostream>

#include <cmath>

template <class T>

T minimum ( T Val1, T Val2) {

if( Val1 > Val2) {

return Val2;

}else {

return Val1;

}

}

template <class T>

T maximum ( T Val1, T Val2) {

if( Val1 < Val2) {

return Val2;

}else {

return Val1;

}

}

template <class T>

T AbsVal(T Val1) {

return abs(Val1);

}

using namespace std;

int main() {

int a = -1, b = 5, e;

double c = -4.5, d = 3.3, f;

cout << "Min" << endl;

e = minimum(a,b);

cout << e << endl;

f = minimum(c,d);

cout << f << endl;

cout << "Max" << endl;

e = maximum(a,b);

cout << e << endl;

f = maximum(c,d);

cout << f << endl;

cout << "AbsVal" << endl;

e = AbsVal(c);

cout << e << endl;

f = AbsVal(a);

cout << f << endl;

return 0;

}

**Output**

Min

-1

-4.5

Max

5

3.3

AbsVal

4

1

Program ended with exit code: 0

**Employee + Production Worker Inheritance**

**main.cpp**

#include <iostream>

#include "ProductionWorker.h"

#include "ShiftSupervisor.h"

int main() {

ProductionWorker PW("Bob", "123134" , "5/5/2016", 1 , 40.5 );

cout << "Production Worker Info" << endl;

cout << "Name: " << PW.getName() << endl;

cout << "ID Number: " << PW.getNum() << endl;

cout << "Hire date: " << PW.getHireDate() << endl;

cout << "Shift: " << PW.getShift() << endl;

cout << "Hourly pay rate: $" << PW.getPayRate() << endl;

ShiftSupervisor SSV("Rob", "123334" , "5/6/2016", 2 , 57.5, 100000, 6000);

cout << "\nShift Supervisor Info" << endl;

cout << "Name: " << SSV.getName() << endl;

cout << "ID Number: " << SSV.getNum() << endl;

cout << "Hire date: " << SSV.getHireDate() << endl;

cout << "Shift: " << SSV.getShift() << endl;

cout << "Hourly pay rate: $" << SSV.getPayRate() << endl;

return 0;

}

**Employee.h**

#ifndef Employee\_h

#define Employee\_h

#include <stdio.h>

#include <string>

using namespace std;

class Employee {

private:

string name;

string number;

string hire\_date;

public:

//Default Constructor

Employee() {

name = "";

number = "";

hire\_date = "";

}

//Constructor

Employee(string emp\_name, string emp\_number, string emp\_hire\_date) {

name = emp\_name;

number = emp\_number;

hire\_date = emp\_hire\_date;

}

~Employee();

string getName() const;

string getNum() const;

string getHireDate() const;

void setName(string N);

void setNum(string EN);

void getHireDate(string HD);

};

#endif /\* Employee\_h \*/

**Employee.cpp**

#include "Employee.h"

Employee::~Employee() {}

string Employee::getName() const {

return this->name;

}

string Employee::getNum() const {

return this->number;

}

string Employee::getHireDate() const {

return this->hire\_date;

}

void Employee::setName(string N) {

this->name = N;

}

void Employee::setNum(string EN) {

this->number = EN;

}

void Employee::getHireDate(string HD) {

this->hire\_date = HD;

}

**ProductionWorker.h**

#ifndef ProductionWorker\_h

#define ProductionWorker\_h

#include <stdio.h>

#include "Employee.h"

#include <string>

class ProductionWorker : public Employee {

private:

int shift;

double hourly\_pay\_rate;

public:

//Constructor & Destructor

//Default Constructor

ProductionWorker ();

//Constructor

ProductionWorker(string PW\_name, string PW\_employee\_number, string PW\_hire\_date, int PW\_shift, double PW\_hourly\_pay\_rate);

~ProductionWorker();

//Accessors & Mutators

int getShift() const;

double getPayRate() const;

void setShift(int S);

void setPayRate(double PR);

};

#endif /\* ProductionWorker\_h \*/

**ProductionWorker.cpp**

#include "ProductionWorker.h"

//Constructor & Destructor

//Default Constructor

ProductionWorker::ProductionWorker () : Employee () {

shift = 0;

hourly\_pay\_rate = 0;

}

//Constructor

ProductionWorker::ProductionWorker(string PW\_name, string PW\_employee\_number, string PW\_hire\_date, int PW\_shift, double PW\_hourly\_pay\_rate)

:

Employee( PW\_name, PW\_employee\_number, PW\_hire\_date) {

shift = PW\_shift;

hourly\_pay\_rate = PW\_hourly\_pay\_rate;

}

ProductionWorker::~ProductionWorker() {}

int ProductionWorker::getShift() const {

return this->shift;

}

double ProductionWorker::getPayRate() const {

return this->hourly\_pay\_rate;

}

void ProductionWorker::setShift(int S) {

this->shift = S;

}

void ProductionWorker::setPayRate(double PR) {

this->hourly\_pay\_rate = PR;

}

**ShiftSupervisor.h**

#ifndef ShiftSupervisor\_h

#define ShiftSupervisor\_h

#include <stdio.h>

#include "ProductionWorker.h"

class ShiftSupervisor : public ProductionWorker {

private:

double annual\_salary;

double bonus;

public:

//Default Constructor

ShiftSupervisor ();

//Constructor

ShiftSupervisor(string SSV\_name, string SSV\_employee\_number, string SSV\_hire\_date, int SSV\_shift, double SSV\_hourly\_pay\_rate, double SSV\_annual\_salary, double SSV\_bonus);

/\*

//Default Constructor

ShiftSupervisor () : ProductionWorker() {

annual\_salary = 70000;

bonus = 3000;

}

//Constructor

ShiftSupervisor(string SSV\_name, string SSV\_employee\_number, string SSV\_hire\_date, int SSV\_shift, double SSV\_hourly\_pay\_rate, double SSV\_annual\_salary, double SSV\_bonus)

:

ProductionWorker ( SSV\_name, SSV\_employee\_number, SSV\_hire\_date, SSV\_shift, SSV\_hourly\_pay\_rate) {

annual\_salary = SSV\_annual\_salary;

bonus = SSV\_bonus;

}

\*/

};

#endif /\* ShiftSupervisor\_h \*/

**ShiftSupervisor.cpp**

#include "ShiftSupervisor.h"

//Default Constructor

ShiftSupervisor::ShiftSupervisor () : ProductionWorker() {

annual\_salary = 70000;

bonus = 3000;

}

//Constructor

ShiftSupervisor::ShiftSupervisor(string SSV\_name, string SSV\_employee\_number, string SSV\_hire\_date, int SSV\_shift, double SSV\_hourly\_pay\_rate, double SSV\_annual\_salary, double SSV\_bonus)

:

ProductionWorker ( SSV\_name, SSV\_employee\_number, SSV\_hire\_date, SSV\_shift, SSV\_hourly\_pay\_rate) {

annual\_salary = SSV\_annual\_salary;

bonus = SSV\_bonus;

}

**Output**

Production Worker Info

Name: Bob

ID Number: 123134

Hire date: 5/5/2016

Shift: 1

Hourly pay rate: $40.5

Shift Supervisor Info

Name: Rob

ID Number: 123334

Hire date: 5/6/2016

Shift: 2

Hourly pay rate: $57.5

Program ended with exit code: 0