

Object Oriented Programming (IGS2130)

Lab 7

Instructor:
Choonwoo Ryu, Ph.D.



INHA UNIVERSITY

Exercise #1



Complete `IntArray` class so that the program can produce the given output

- Copy constructor
- `set()`

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

class IntArray {
private:
    int* m_data;
    int m_len;
public:
    IntArray(int = 0, int = 0);
    ~IntArray();
    void print(void);
};

IntArray::IntArray(int size, int init) {
    if (size <= 0) {
        m_data = nullptr; m_len = 0;
    }
    else {
        m_data = new int[size];
        m_len = size;
        for (int idx = 0; idx < m_len; ++idx)
            *(m_data + idx) = init;
    }
}
```

```
IntArray::~IntArray() {
    delete[] m_data;
}

void IntArray::print(void) {
    for (int idx = 0; idx < m_len; ++idx)
        cout << *(m_data + idx) << ' ';
    cout << std::endl;
}

int main() {
    cout << "a1: ";
    IntArray a1{ 10, 100 };
    a1.print();

    cout << "a2: ";
    IntArray a2{ a1 };
    a2.set(3, 999);
    a2.set(9, 123);
    a2.print();

    return 0;
}
```

```
a1: 100 100 100 100 100 100 100 100 100 100
a2: 100 100 100 999 100 100 100 100 100 123
```

Exercise #2



- Create **three friend functions** of **IntArray** class so that the following program can produce the given output

- reverse()
- negative()
- multiply()

```
int main() {  
    IntArray a1{ 5, 0 };  
    for (int idx = 0; idx < 5; ++idx)  
        a1.set(idx, 100 +idx*50);  
    cout << "a1:          "; a1.print();  
  
    reverse(a1);  
    cout << "reverse: "; a1.print();  
    multiply(a1, 3);  
    cout << "multiply: "; a1.print();  
    negative(a1);  
    cout << "negative: "; a1.print();  
  
    return 0;  
}
```

```
a1:    100 150 200 250 300  
reverse: 300 250 200 150 100  
multiply: 900 750 600 450 300  
negative: -900 -750 -600 -450 -300
```

Exercise #3: OOP Project: Step 04



- Upgrade our bank application version 0.3 to 0.4
 - Account class
 - Separate class definition and member function definitions

- Create Account handler class named **AccountHandler**
 - Manage the **Account** class objects
 - Include Account pointer array as a member variable

```
Account* accArr[MAX_ACC_NUM]; // Account array
int accNum = 0;                // # of accounts
```

- Include all non-member function as member functions

```
void ShowMenu(void);
void MakeAccount(void);
void DepositMoney(void);
void WithdrawMoney(void);
void ShowAllAccInfo(void);
int GetAccIdx(int);
```

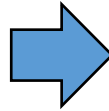
- Modify main() function so it can run with the **AccountHandler** object

Separation of class definition and member function definitions



```
#include <iostream>
using namespace std;
class Date {
private:
    int m_year;
    int m_month;
    int m_day;

public:
    void show() {
        cout << m_year << "/"
        << m_month << "/" << m_day << endl;
    }
};
```



```
#include <iostream>
using namespace std;
class Date {
private:
    int m_year;
    int m_month;
    int m_day;

public:
    void show();
};

void Date::show() {
    cout << m_year << "/" << m_month
    << "/" << m_day << endl;
}
```

Definition of Account classes



Account class

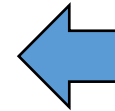
```
class Account {  
private:  
    int m_accID;  
    int m_balance;  
    char* m_cusName;  
  
public:  
    Account(const Account& ref);  
    Account(int ID, int balance, char* cname);  
    ~Account();  
    int GetAccID(void) const;  
    void Deposit(int money);  
    int Withdraw(int money);  
    void ShowAccInfo(void) const;  
};
```

Definition of AccountHandler class

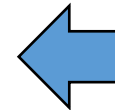


AccountHandler class

```
class AccountHandler {  
private:  
    Account* m_accArr[MAX_ACC_NUM];  
    int m_accNum;  
  
    int GetAccIdx(int id) const;  
public:  
    AccountHandler();  
    ~AccountHandler();  
    void ShowMenu(void) const;  
    void MakeAccount(void);  
    void DepositMoney(void);  
    void WithdrawMoney(void);  
    void ShowAllAccInfo(void) const;  
};
```



```
Account* accArr[MAX_ACC_NUM];  
int accNum = 0;
```



```
void ShowMenu(void);  
void MakeAccount(void);  
void DepositMoney(void);  
void WithdrawMoney(void);  
void ShowAllAccInfo(void);  
int GetAccIdx(int);
```

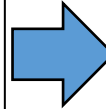
main() function



```
int main(void) {
    int choice, i;

    while (1) {
        ShowMenu();
        cout << "Select menu: ";
        cin >> choice;
        cout << endl;

        switch (bank(choice)) {
            case bank::MAKE:
                MakeAccount();
                break;
            case bank::DEPOSIT:
                DepositMoney();
                break;
            case bank::WITHDRAW:
                WithdrawMoney();
                break;
            case bank::INQUIRE:
                ShowAllAccInfo();
                break;
            case bank::EXIT:
                for (i = 0; i < accNum; i++)
                    delete accArr[i];
                return 0;
            default:
                cout << "Illegal selection.." << endl;
        }
    }
    return 0;
}
```



```
int main(void) {
    AccountHandler manager;
    int choice;
    bool run = true;

    while (run) {
        manager.ShowMenu();
        cout << "Select menu: ";
        cin >> choice;
        cout << endl;

        switch (bank(choice)) {
            case bank::MAKE:
                manager.MakeAccount();
                break;
            .....
            default:
                cout << "Illegal selection.." << endl;
        }
    }
    return 0;
}
```