C언어 기반의 C++ 2

2주차

C언어 복습

- const의 의미
  - const int num = 10; // num의 값을 바꿀 수 없음
  - const int *ptr1 = &val1; // ptr1의 역참조 값을 바꿀 수 없음.
  - int * const ptr2 = &val1; // ptr2가 다른 주소를 가질 수 없음.

- 실행중인 프로그램의 메모리 공간
  - 데이터 : 전역변수, static 변수
  - 스택 : 지역변수 (매개변수)
  - 힙 : malloc으로 동적할당 (free로 되돌려 줌)

- 함수 호출 시 인자 전달 방법
  - Call-by-value : 매개변수에 값이 복사됨
  - Call-by-reference : 포인터를 매개변수로 주어 주소값을 복사하게 함으로써 호출하는 쪽과 메모리를 공유할 수 있음.

참을 의미하는 true, 거짓을 의미하는 false (TrueAndFalse.cpp)

64-66p

```
#include <iostream>
using namespace std;
int main(void) {
        int num = 10;
        int i = 0;

        cout << "true: " << true << endl;
        cout << "false: " << false << endl;
        while (true) {
                cout << i++ << ' ';
                if (i > num)
                        break;
        }
        cout << endl;
        cout << "sizeof 1: " << sizeof(1) << endl;
        cout << "sizeof 0: " << sizeof(0) << endl;
        cout << "sizeof true: " << sizeof(true) << endl;
        cout << "sizeof false: " << sizeof(false) << endl;
        return 0;
}
```

true는 '참', false는 '거짓'을 의미하는 1바이트 데이터.
이 둘은 각각 정수 1과 0이 아님.
정수로 변환하면 각각 1과 0으로 보일 뿐.

자료형 bool의 활용

- **true와 false는 bool형 데이터**
- **bool 도 기본자료형**

```
#include <iostream>
using namespace std;
bool IsPositive(int num) {
        if (num < 0) return false;
        else return true;
}
int main(void) {
        bool isPos;
        int num;
        cout << "Input number: ";
        cin >> num;
        isPos = IsPositive(num);
        if (isPos)
                cout << "Positive number" << endl;
        else
                cout << "Negative number" << endl;
        return 0;
}
```

Reference의 이해

◆ **Reference : 다른 이름을 추가로 주는 것.**

- **변수에 대해서만 reference 만들 수 있음**
- **Reference는 선언문에서 초기화 해야 함**
- **Reference의 reference도 만들 수 있음.**
- **한 변수의 reference를 여러 개 만들 수도 있음.**

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

int main(void) {
        int num1 = 1020;
        int& num2 = num1;

        num2 = 3047;
        cout << "VAL: " << num1 << endl;
        cout << "REF: " << num2 << endl;

        cout << "VAL: " << &num1 << endl;
        cout << "REF: " << &num2 << endl;
        return 0;
}
```

변수 num1과 그의 reference num2의 값과 주소가 똑같게 나온다.

배열 원소와 포인터의 Reference
(RefArrElem.cpp, RefPtr.cpp)

```
int main(void) {
        int arr[3] = { 1, 3, 5 };
        int& ref1 = arr[0]; int& ref2 = arr[1]; int& ref3 = arr[2];

        cout << ref1 << endl; cout << ref2 << endl; cout << ref3 << endl;

        int num = 12;
        int* ptr = &num;
        int** dptr = &ptr;

        int& ref = num;
        int* (&pref) = ptr;
        int** (&dpref) = dptr;

        cout << ref << endl;
        cout << *pref << endl;
        cout << **dpref << endl;
        return 0;
}
```

배열 원소에 대한 reference가능.
배열에 대해서는 불가능.

포인터에 대한 reference 가능.

참조자를 이용한 call-by-reference
(RefArrElem.cpp, RefPtr.cpp)

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

void SwapByRef2(int &ref1, int& ref2)
{
        int temp=ref1;
        ref1=ref2;
        ref2=temp;
}

int main(void)
{
        int val1=10;
        int val2=20;

        SwapByRef2(val1, val2);
        cout<<"val1: "<<val1<<endl;
        cout<<"val2: "<<val2<<endl;
        return 0;
}
```

호출되는 시점에
main함수의 val1, val2의 다른 이름으로
SwapByRef2 함수의 ref1, ref2개 생긴다.

[문제1] 참조자 이용 함수 작성

- 인자로 전달된 int 형 변수의 값을 1 증가시키는 함수
- 인자로 전달된 int 형 변수의 부호를 바꾸는 함수
- 위 두 함수를 호출하는 main 함수

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

void IncreOne(int& num) { num++; }
void InverSign(int& num) { num *= -1; }

int main(void)
{
	int val = 20;
	IncreOne(val);
	cout << val << endl;
	InverSign(val);
	cout << val << endl;
	return 0;
}
```

81-82p

함수매개변수 참조자의 적절한 const 사용

- 함수 내에서 수정하지 않는 참조변수는 const로 만들어 side effect 없음을 명확히 하자!
  - void HappyFunc(const int &ref) { … }

87-88p

함수반환형으로서의 참조형

- 인자로 받은 참조형을 반환할 수 있음.
- 함수 내의 지역변수에 대한 참조변수를 반환하면 안됨

함수반환형이 참조형일 때 (RetReturnOne.cpp)

- 인자로 받은 참조형을 반환할 수 있음.
- 호출하는 쪽에서 참조형 변수로 받으면 새로 생기지 않고 기존 변수에 이름만 추가되는 것,
  일반 변수로 받으면 새 변수가 생성되어 값이 복사되는 것.
  (함수 매개변수로 참조자 넘길 때와 같은 현상)

```
#include <iostream>
using namespace std;
int& RefRetFuncOne(int& ref) {
        ref++;
        return ref;
}
int main(void) {
        int num1 = 1;
        int& num2 = RefRetFuncOne(num1);
        num1++;
        num2++;
        cout << "num1: " << num1 << endl;
        cout << "num2: " << num2 << endl;
        return 0;
}
```

참조변수라도 일반형인 int 로 반환할 수 있음.

이 3개가 모두 같은 변수의 다른 이름.

일반형인 int 로 받으면 새로 변수 생기고 값 복사됨.

const 매개변수의 다른 의미

- 상수화된 변수
  - 선언 방법 : const int &ref = 30;
  - 내부 처리 방법 : 임시변수 생성하여 30을 보관하고 ref 로써 참조.

- 상수화된 변수를 이용한 const 참조 매개변수
  - 함수 선언 방법
    int Adder(const int &num1, const int &num2) { … }
  - 함수 호출 방법
    cout << Adder(3,4) << endl;
  - 매개변수에 const가 없다면 참조변수 자리에 상수를 줄 수는 없다.

[문제2] const 포인터와 const 참조자

- **포인터 변수를 선언하여 아래 변수를 가리켜 보자**
  const int num = 12;
- **위 포인터 변수에 대한 참조변수를 선언해 보자.**
- **포인터 변수와 참조변수를 이용하여 num의 값을 한번씩 출력하자.**

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

int main(void)
{
	const int num = 12;
	const int* ptr = &num;
	const int* (&ref) = ptr;

	cout << *ptr << endl;
	cout << *ref << endl;

	return 0;
}
```

malloc, free -> new, delete

- malloc, free 방식의 2가지 단점 해결
  - Byte 단위로 크기계산 해야 함
  - 반환 값에 대한 명시적인 형변환 필요

- new 방법
  배열을 받을 때는 [] 안에 개수 표시.
  - int *ptr1 = new int;
  - double *ptr2 = new double;
  - int *arr1 = new int[3];
  - double *arr2 = new double[7];

- delete 방법
  배열을 반환할 때는 [] 만 표시.
  - delete ptr1;
  - delete ptr2;
  - delete [] arr1;
  - delete [] arr2;

- 객체 생성에는 반드시 new 사용! (malloc 금지)

포인터를 사용하지 않고 힙에 접근하기

- new 연산자로써 할당된 메모리 공간에 대해서도 참조자 생성 가능
  - 포인터 연산 없이 heap 사용 가능

```
int *ptr = new int;
int &ref = *ptr;
ref = 20;
cout << *ptr << endl;
```

[문제3] 구조체에 대한 new, delete

- 좌표값을 표현하는 구조체가 있다.

```
typedef struct {
    int xpos;
    int ypos;
} Point;
```

- 아래 형태의 함수를 정의하라. 이 함수는 할당 받은 구조체를 반환한다.

```
Point &PointAdder( const Point &p1, const Point &p2 );
```

- 위 함수를 호출하는 main 함수를 작성하라. 단, main 함수에서 구조체 변수는 선언이 아닌 new 에 의한 할당으로만 한다.
- 할당 받은 메모리는 사용이 끝난 후 반환하라.

[문제3] 구조체에 대한 new, delete

```
#include <iostream>
using namespace std;
typedef struct {
        int xpos;
        int ypos;
} Point;
Point &PointAdder(const Point & p1, const Point & p2) {
        Point * pptr = new Point;
        pptr->xpos = p1.xpos + p2.xpos;
        pptr->ypos = p1.ypos + p2.ypos;
        return *pptr;
}
int main(void) {
        Point * pptr1 = new Point; pptr1->xpos = 3; pptr1->ypos = 30;
        Point * pptr2 = new Point; pptr2->xpos = 70; pptr2->ypos = 7;

        Point & pref = PointAdder(*pptr1, *pptr2);
        cout << "[" << pref.xpos << ", " << pref.ypos << "]" << endl;

        delete pptr1; delete pptr2; delete & pref;
        return 0;
}
```

타입을 맞춤.

C++에서 C언어의 표준함수 호출하기

- C언어의 표준 헤더파일과 대응되는 C++ 표준 헤더 파일 사용하려면
  (1) 앞에 c 붙이고
  (2) 뒤의 .h 생략한 이름을 include하고 사용.
  - 예: #include <stdio.h>
    -> #include <cstdio>

- C언어의 표준 헤더파일도 사용할 수 있지만 함수 overloading 등의 차이가 있으므로 C++ 표준 헤더를 사용하자.

[문제4] C++ 표준함수 호출

◆ **다음 3 함수를 이용하여 5 이상 200 이하의 난수 5개를 출력하는 프로그램을 작성하시오. C언어에서는 time 함수는 <time.h>에, rand와 srand 함수는 <stdlib.h>에 선언되어 있으나 여기서는 C++ 헤더를 사용하여 작성하자.**

- **rand, srand ,time**

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

int main(void)
{
    srand(time(NULL));
    for (int i = 0; i < 5; i++)
        printf("Random number #%d: %d₩n", i, (rand() % 196) + 5);
    return 0;
}
```

[실습] 컴퓨터와 가위바위보

- **사용자와 컴퓨터가 가위바위보를 진행하게 한 후 판정해 주는 프로그램을 작성하시오. 사용자의 선택을 입력 받고 컴퓨터의 선택은 임의의 수로 선택하시오.**

[결과]
가위(1) 바위(2) 보(3) : 1
당신이 낸 것 : 가위
컴퓨터가 낸 것 : 바위
컴퓨터가 이겼다!

[실습] 컴퓨터와 가위 바위보

```
#include <iostream>
#include <ctime>
using namespace std;
int main(void) {
	const char* name[3] = { "가위", "바위", "보" };
	int you, computer;

	srand(time(NULL));
	cout << "가위(1) 바위(2) 보(3) : ";
	cin >> you;
	you--;
	computer = rand() % 3;

	cout << "당신이 낸 것 : " << name[you] << endl;
	cout << "컴퓨터가 낸 것 : " << name[computer] << endl;
	int prev_you = (you + 2) % 3;
	int next_you = (you + 1) % 3;
	if (computer == next_you)
		cout << "컴퓨터가 이겼다!" << endl;
	else if (computer == prev_you)
		cout << "당신이 이겼다!" << endl;
	else
		cout << "비겼다!" << endl;
	return 0;
}
```